

Streamlining
C++ Programs

Price \$4.50 U.S.
(Canada \$5.95)

The Users Journaltm

March 1994

ID: IC4
Lichtman's
\$5.95

12, Number 3

Advanced Solutions for C/C++ Programmers

Real-Time/ Embedded Systems

- Fuzzy Logic Replaces a PID Loop
- Symbolic Control for Embedded Systems
- Embedding an EXE File

Also:

- Making OOP
More Efficient
- Adding Prediction
to the Korn Shell
- Book Review:
C Elements of Style

Columns By:

P.J. Plauger

Dan Saks

Chuck Allison

Ken Pugh

Victor R. Volkman



0 38808 74098 9

We're Talking Major Flexibility Here

Greenleaf can control, store, compress, Windowize, and organize your data...
That's just the beginning: read more about the products that do it...

Greenleaf ArchiveLib

- ✓ Supports Windows 3.1, MS-DOS, and can be ported to OS/2 and UNIX.
- ✓ No more spawning calls to data compression programs from within your application: all you do is call ArchiveLib functions from your C or C++ application.
- ✓ You hold the reins for managing what is compressed or expanded, in what order, where it is placed, how it is retrieved, etc.
- ✓ Lets you specify the source and destination for any media (file, buffer, serial stream, parallel port, etc.)
- ✓ Windows 3.1 DLL and static link libraries included (DLL is linkable with most DLL-capable compilers including Visual Basic).
- ✓ Supports Borland, Microsoft, Symantec, TopSpeed, Visual C++ and WATCOM C/C++ compilers.
- ✓ BUILD utility will rebuild the library for your specific compiler.
What about the competition? Here's how ArchiveLib stacks up:
- ✓ The competition's data compression package contains two functions; ArchiveLib contains over 4 classes and 100 functions (for C and C++ programmers).
- ✓ The competition simply compresses data; ArchiveLib compresses ASCII or binary data and archives it.

Greenleaf Database Library

- ✓ Supports Windows 3.1, MS-DOS, and can be ported to OS/2 and UNIX.
- ✓ Over 150 C functions access industry standard database data, index, and memo files at an affordable price.
- ✓ Windows 3.1 DLL and static link libraries included (DLL is linkable with most DLL-capable compilers including Visual Basic).
- ✓ Unsurpassed speed and flexibility let you interface with dBASE III, dBASE IV, FoxPro, FoxBASE, Clipper, dBase, and other xBASE files including new FoxPro CDX index files.
- ✓ Does not require any database package: product is a complete ISAM library.
- ✓ Lets you use index and/or memo file types other than the ones associated with your database.
- ✓ Unlimited number of record and file locks available, make it fully compatible with multiuser and network applications.
- ✓ Single and multiple user network access fully supported.
- ✓ Supports Borland, Microsoft, Symantec, TopSpeed, Visual C++ and WATCOM C/C++ compilers.
- ✓ BUILD utility will rebuild the library for your specific compiler.

Greenleaf ArchiveLib v1.0

\$279

Greenleaf Database Library v3.22

\$249

- ☐ FREE Source code--all of it!
- ☐ FREE Unlimited Tech Support.
- ☐ FREE online Windows and DOS help system.
- ☐ Compilable examples available in three places: online help, *Reference Manual*, and on disk
- ☐ FREE access to Greenleaf BBS.
- ☐ Professional quality, same day shipment, personalized service.
- ☐ Greenleaf Gold Support is also available--ask about it!
- ☐ 60-day money-back guaranty (if source code not opened).



Call Greenleaf Software today for complete information or to order. Major credit cards, COD, approved purchase orders. Same day shipment in most cases.

(800)523-9830

(214)248-2561 FAX: (214)248-7830

BBS: (214)250-3778 for free information.

16479 Dallas Parkway, Suite 570, Dallas, TX 75248



Can't Wait To See In UNIX.

Do Windows Without Getting Soaked!

It's an eyeopener! The multi-user, multi-tasking power of UNIX in Coherent with X-Windows. It's the GUI everyone's looking for. Multiple overlapping windows. Graphic color. And the price? You won't believe your eyes!

But Coherent was independently developed by Mark Williams Company as a virtual clone of UNIX. So it comes with a very un-UNIX-like price.

Run The World's Best Selling Software!

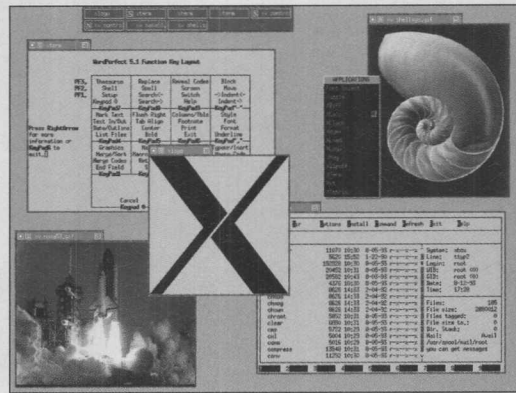
Coherent now runs

WordPerfect, Lotus 1-2-3 and most every other UNIX PC application right out of the box. Call for current information. Whatever the software, chances are Coherent runs it now!

Small Is Beautiful In So Many Ways.

Coherent embraces the original UNIX philosophy: Small is beautiful. Small price, obviously. But Coherent has other size advantages as well.

Its 5-disk installation is a breeze. Just plug and play. No need for CD-ROM drives or special device adapters. And Coherent can reside with DOS so you can keep all your DOS windows applications, too.



Software Critics And Over 60,000 Users Say it All!

Coherent has been hailed as possibly the best thing that ever happened to UNIX. And with tens of thousands of users, the critics are still marveling.

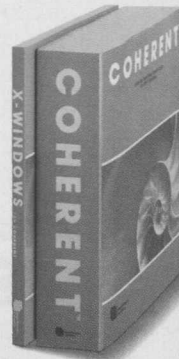
"For price/performance, Coherent beats every other 32-bit operating system in the business."
—Steven J. Vaughan-Nichols, Computer Shopper

"To paraphrase Annie Oakley, 'Anything U(NIX) can do, Coherent can do better.'
—Michele Petrovsky, LAN Computing

Coherent With X-Windows

\$149.95

(Without X-Windows, \$99.95)



Don't Let The Price Fool You

This is no stripped down UNIX wannabe. Coherent is a fully professional multi-user system with a complete C compiler, assembler and over 200 powerful UNIX commands for development, administration, maintenance and text processing. You could pay thousands for all this. Plus you get our acclaimed 1200-page manual.

You Simply Can't Go Wrong.

Mark Williams Company has been developing professional programming products since 1976. Our free unlimited technical support along with our popular BBS are renowned. We ship most orders within 24 hours, which means you'll have the power of Coherent in your hands in no time. And with our extended money-back guarantee, you can't miss.

So pick up the phone and order Coherent 4.2 with X-Windows today. And get everything you can't wait to see.



Mark Williams Company
60 Revere Drive
Northbrook, IL 60062

Order Now! 1-800-636-6700

or 708-291-6700, FAX: 708-291-6750

Coherent is a trademark of Mark Williams Company. UNIX is a trademark of USL. All other product and company names mentioned are trademarks or registered trademarks of their respective companies.

Distributors: Australia + 07-266-2270, Brazil + 011-883-2299, Czech. + 06-32-62877, France +1-4741-4519, + 1-4672-8074, Germany + 0511-53-7295, Greece + (01) 222-3511, Holland + 2240-18499, Hungary + 153-0988, Malaysia + 03-756-4477, Venezuela/Caribbean + 1-809-723-5000, Sweden + 06-60-19290, UK + (0) 81-541-5466

Periodic Table of Visual Components

5

C++

Visual C++ 1.5

Component Number

Symbol

Proper Name

IIIB

IVB

VB

5

C++

Visual C++ 1.5

6

BOOKS

Books Online

7

CCMD

CCmdTarget

13

APP

App Studio

14

MFC

Microsoft
Foundation Class
Library 2.5

15

CWIN

CWinApp

VIIIA

IB

IIB

27

CCB

CControlBar

28

OLE

Object Linking &
Embedding 2.0

29

ODBC

Open Database
Connectivity

30

DDX

Dialog Data
Exchange

31

APW

AppWizard

32

CLW

ClassWizard
Controls

33

CDOC

CDocument

45

CTB

46

COLE

ColeServer

47

CREC

CRecordSet

48

DDV

Dialog Data
Validation

49

HELP

50

CFRM

CFormView

51

CWNE

CWne

82

83

THE BUILDING BLOCKS OF VISUAL DEVELOPMENT.

The elements of next generation applications are found in the components of the Microsoft® Visual C++™ development system, version 1.5 today.

Innovative *wizard* technology allows you to seamlessly meld components into sophisticated applications, in weeks not months. Just remember this simple equation: MFC 2.5, OLE 2.0, ODBC.

That's a perfect formula for building and reusing components. At the nucleus is MFC 2.5 (Microsoft Foundation Class Libraries), the standard API for Windows.™ The new OLE (Object Linking and Embedding) classes make it easier than ever to implement OLE's unprecedented integration power, new drag-and-drop features, visual editing and OLE automation. In all, it provides more than 19,000 lines of code you want, but won't have to write or rewrite.

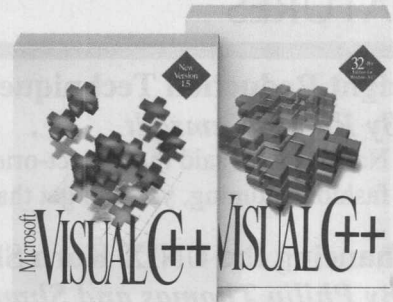
In addition, the new ODBC (Open Database Connectivity)

classes provide access to major databases and data binding without coding. It's a powerful new platform for creating high-performance database applications.

Best of all, your 16-bit MFC applications are portable to the Windows NT™ operating system and future versions of Windows. Now that's good chemistry.

If you want the latest tools for the latest Windows technology, ask for Microsoft Visual C++ 1.5.

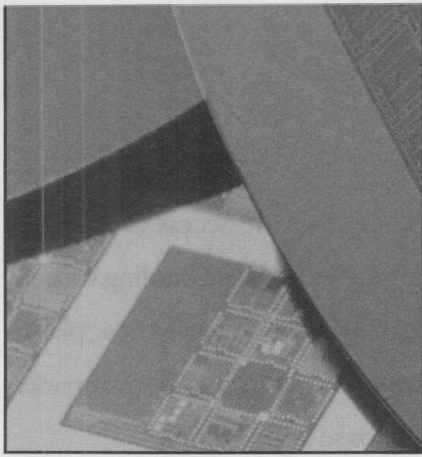
Call us now at (800) 434-3980 and we'll make you a catalyst for change today and into the future of Windows operating systems.



New 16-bit version 1.5 joins the Visual C++ family of development systems for Windows and Windows NT.

Microsoft®

© 1993 Microsoft Corporation. All rights reserved. In the 50 United States, call (800) 434-3980; customers in Canada, call (800) 563-9048; outside the U.S. and Canada, call your local Microsoft subsidiary or (206) 936-8661. Microsoft is a registered trademark and Visual C++, Windows and Windows NT are trademarks of Microsoft Corporation.



The **C** Users Journaltm

Advanced Solutions for C/C++ Programmers

REAL-TIME/EMBEDDED SYSTEMS

Symbolic Access to Embedded Controllers

- By Odd A. S. Olsen and Petter H. Heyerdahl* **21**
The principle of deferred binding even applies to embedded systems sometimes.

An Embedded System EXE Locator

- By Charles B. Allison* **35**
Here's an important missing ingredient in turning your PC compiler into a development platform for embedded systems.

A Fuzzy-Logic Torque Servo

- By Jack J. McCauley* **47**
Intrigued by the promise of fuzzy logic? Here's a simple real-world application that shows how the promise can pay off.

FEATURES

Weight Reduction Techniques in C++

- By Randy Kamradt* **70**
Nobody ever said that object-oriented programming gives you something for nothing. But with a little old-fashioned tuning, you can get that something and keep good performance in the bargain.

Enhancing the UNIX Korn Shell Using Predictor Techniques

- By Philip Thomas and Shmuel Rotenstreich* **83**
If you like a shell that keeps a command history, you'll love one that predicts the future as well.

BOOK REVIEW

- C Elements of Style* *Reviewed by Dwayne Phillips* **115**

Cover Photo by Steve Dunwell © 1987 Steve Dunwell/The Image Bank

The C Users Journal (ISSN 0898-9788) is published monthly by R&D Publications, Inc., 1601 W. 23rd St., Suite 200, Lawrence, KS 66046-2700 (913) 841-1631. Second-class postage paid at Lawrence, KS and additional mailing offices. POSTMASTER: Send address changes to THE C USERS JOURNAL, 1601 W. 23rd St., Suite 200, Lawrence, KS 66046-2700. Subscriptions: Annual renewable subscriptions to The C Users Journal are \$29.95 US, \$46 Canada and Mexico, \$65 overseas. Payments must be made in US dollars. Make checks payable to The C Users Journal. GST (Canada): #129065819

COLUMNS

Standard C: C++ Language Support Library

P. J. Plauger 10

Questions & Answers: Run-Time Type Checking in C++

Kenneth Pugh 65

Stepping Up to C++: The Return Types of Virtual Functions

Dan Saks 91

Code Capsules: The Preprocessor

Chuck Allison 101

CUG New Releases:

IOCCC, ASXXXX, MINED, TDE Update, and a Bug Fix

Victor R. Volkman 117

CUG Product Focus: C++ SIM

Victor R. Volkman 119

DEPARTMENTS

Editor's Forum 8

Product Pages 79

Advertiser Index 112

Calendar of Events 126

New Products 131

InstantInfo Index 135

Call for Papers 136

We Have Mail 137

Programmer's Market 138

CUG Online Source Code

You can obtain all code published (and some unpublished) in *CUG* from:

USENET (Archived by UUNET Technologies):

uunet!~/published/cuj/19YY/monYY.tar.Z

Available via anonymous FTP from *ftp.uu.net* or via *uucp* from (900)GOT-SRCS.

Download via *uucp* and uncompress using *compress-d filename*, then *tar xvf filename* to expand the archive.

BBS:

CornerStone, 206-362-4283; The Courts of Chaos, 501-985-0059; EmmaSoft Shareware Board, 607-533-7072; Phoenix Chapter ACM Library, 602-970-0474; The Programmer's Corner, 301-596-7692; The Brass Cannon, 801-226-8310.

CompuServe:

In Library 7, CLMFORUM

GEnie:

In the IBM-PC Roundtable at page 615 (Keyword: IBMPC).

Monthly Code Disk:

Call R&D Publications, Customer Relations, (913) 841-1631.

The C Users Journal™

EDITORIAL

Publisher

Robert Ward

Senior Editor

P. J. Plauger

Managing Editor

Marc Briand

Contributing Editors

Chuck Allison

Steve Graham

Kenneth Pugh

Dan Saks

Victor R. Volkman

Sydney Weinstein

CUSTOMER SERVICE

Customer Service

Pam VanSchmus

Jodi Leonard

913-841-1631

FAX 913-841-2624

ADVERTISING AND MARKETING

Marketing Director

Jeff Dickey-Chasins

Acct. Manager, East

Ed Day

913-841-1622

Acct. Manager, Midwest

Christine Woodley

913-841-6733

Acct. Manager, West

Edwin Rothrock

913-841-1626

European

Brian Osborn

Advertising

Hohenleuchte 10 A

Representative

D-24159 Kiel

Germany

+49 431-396895

FAX +49 431-396827

Sales Services

Danelia Quiroz

Advertising Services

Noelle M. Martin

Direct Marketing

Bill Uhler

PRODUCTION

Art Director

Susan Schuette Buchanan

Graphic Artist

Twyla Watson Bogaard

Production Editor

Liza Behymer

Advertising Materials

Lori White

The C Users Journal is the successor to the *C Users' Group Newsletter* and *The C Journal*. Subscribers are automatically enrolled as members of The C Users' Group. The *C Users Journal* and The C Users' Group are services of R&D Publications, Inc., Lawrence, KS.

Entire contents Copyright ©1994 R&D Publications, Inc. No portion of this publication may be reproduced, stored, or transmitted in any form, including computer retrieval, without written permission from the publisher. All Rights Reserved. Quantity reprints of selected articles may be ordered. By-lined articles express the opinion of the author and are not necessarily the opinion of the publisher.

Printed in the United States of America.

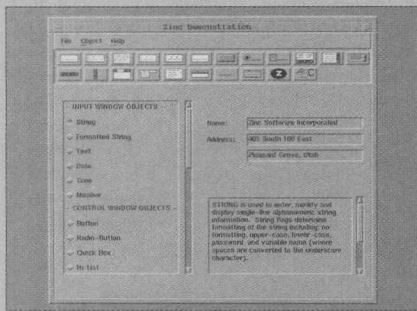


Advertising: For rate cards or other information on placing advertising in *The C Users Journal*, contact the advertising department at (913) 841-1631, or write *The C Users Journal*, 1601 W. 23rd St., Ste. 200, Lawrence, KS 66046-2700.

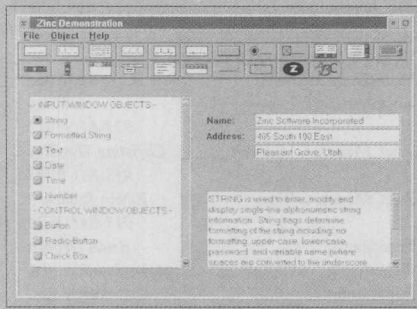
Customer Service: For subscription orders and address changes, contact *The C Users Journal*, 1601 W. 23rd St., Ste. 200, Lawrence, KS 66046-2700. Telephone (913) 841-1631; FAX (913) 841-2624; e-mail cujsub@dpub.com.

Trademarks: The C Users Journal, R&D Publications, Inc. UNIX, AT&T Bell Laboratories. XENIX, MS-DOS, OS/2, Microsoft C and QuickC, Windows, Microsoft Corporation. IBM, IBM-PC, PS2, International Business Machines Corp. Brief, Turbo C, Turbo C++, Borland International. Macintosh, Apple Computer, Inc. Zortech C++, Zortech. VAX-VMS, Digital Equipment Corp., DR-DOS, Digital Research. DESQview, Quarterdeck. Atari, Atari, Inc., Amiga, Commodore, Inc.

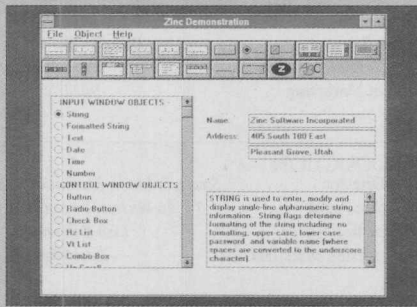
THINK
ZINC



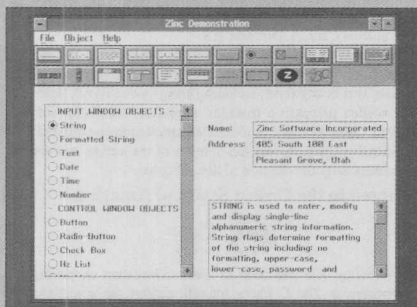
UNIX MOTIF



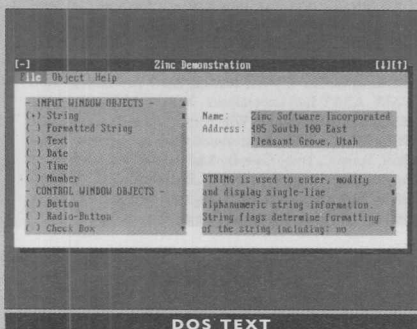
OS / 2 2.0



MS WINDOWS & WINDOWS NT



DOS GRAPHICS



DOS TEXT

ZINC™ APPLICATION FRAMEWORK™ 3.5

Multiplatform Flexibility From One Application Framework.

FLEXIBILITY is an essential component in software design. Your development tools should allow you to easily incorporate new features and technologies into your applications without rewriting all of your code. That's a tall order if your development tools are designed like a straight-jacket. Zinc Application Framework 3.5 and Zinc Designer™ give you flexible and extendible support for Microsoft Windows, Windows NT, OS/2 2.0, UNIX Motif, DOS Graphics and DOS Text. And Zinc does it with ONE set of source code. Zinc's multiplatform, object-oriented architecture won't confine your application development options today...or tomorrow.

FOR A FREE ZINC DEMO KIT, CALL US TOLL FREE TODAY AT

1.800.638.8665. IN EUROPE CALL +44 (0) 81 855 9918

Z

i

n

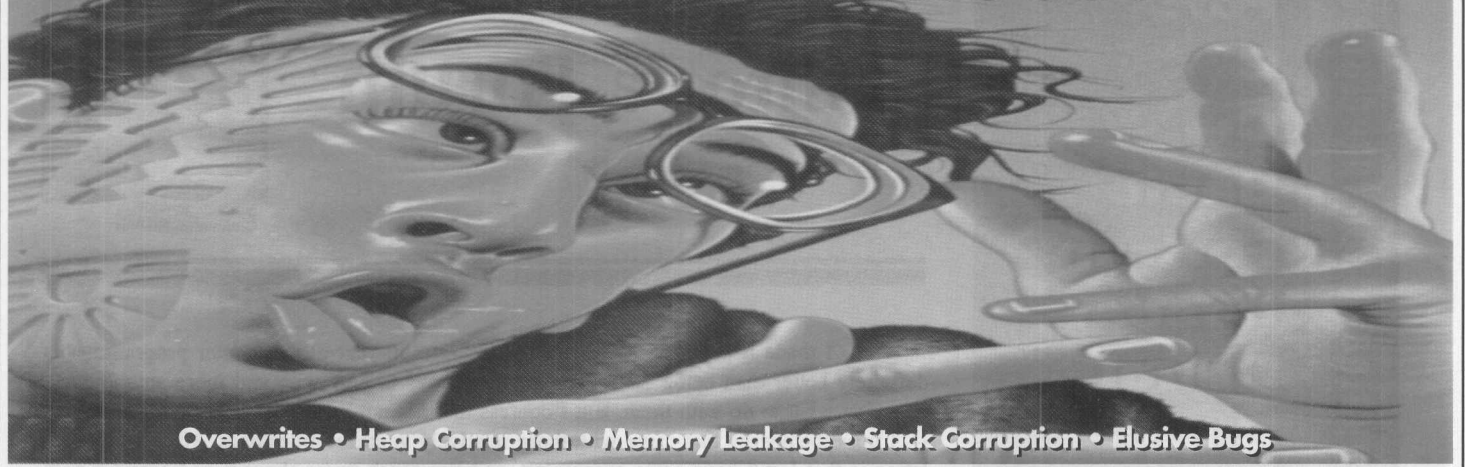
c

ZINC SOFTWARE INCORPORATED, 405 SOUTH 100 EAST, 2ND FLOOR, PLEASANT GROVE, UTAH 84062, TEL 801-785.8900, FAX 801-785.8996, BBS 801-785.8997.

EUROPE: ZINC SOFTWARE (UK) LIMITED 58-60 BERESFORD STREET, LONDON SE18 6BG, TEL +44 (0) 81 855 9918, FAX +44 (0) 81 316 7778, BBS +44 (0) 81 317 2310

© COPYRIGHT 1992 ZINC SOFTWARE INCORPORATED, ALL RIGHTS RESERVED. ZINC, ZINC DESIGNER AND ZINC APPLICATION FRAMEWORK ARE TRADEMARKS OF ZINC SOFTWARE INCORPORATED. OTHER TRADEMARKS ARE OWNED BY THEIR RESPECTIVE COMPANIES.

STILL TURNING THE OTHER CHEEK TO THE SAME OLD BUGS?



Overwrites • Heap Corruption • Memory Leakage • Stack Corruption • Elusive Bugs

Introducing MemCheck 3.0 for DOS

With this **major new release**, MemCheck once again establishes itself as the **number one** defense against the worst bugs in C. MemCheck 3.0 is a **quantum leap** in bug-stomping power and is **super-tuned for blazing speed** that just plain leaves its competition in the dust. MemCheck 3.0 now detects errors **in every part of your application**, even in **third-party libraries** for which you don't have the source code! And unlike other tools that try to do some of the same checking, MemCheck works **perfectly** with your favorite debugger, and can even be used to find problems at **your customers' sites!**

**NEW
VERSION
3.0**

Find hidden bugs

You know how hard it is to write the **perfect C program**. Leave **one little detail** out of place, and you'll spend **hours, days, or even weeks** looking for what went wrong. Errors can be as simple as forgetting to free allocated memory, or as elusive as overwriting the end of a block **by a single byte**. So if you're serious about writing **rock-solid applications**, get MemCheck: **the fastest, easiest defense** against a whole cadre of memory bugs.

Engineered for ease of use

Best of all, whatever platform you use it on, MemCheck does its work **silently and automatically**. You'll get **concise, informative messages** that pinpoint what's wrong, usually with the **exact source file and line number**. Like unfreed memory, or overwritten buffers. Null pointer assignments, invalid calls to `free()`, destructive `strcpy()`s, `memcpy()`s, and more. With its **automatic operation** and **low profile**, MemCheck will prove to be one of the most valuable tools in your development arsenal.

MEMCHECK PRICING

Be sure to **specify compiler and platform** when ordering

MEMCHECK 3.0 FOR DOS \$139
Specify Microsoft C, or Borland C, or Watcom C
(Upgrade from MemCheck 2.1 \$79)

MEMCHECK 2.1 FOR WINDOWS \$179 \$99
Specify Microsoft C or Borland C
MEMCHECK 2.1 FOR THE MAC \$179 \$99
Specify Think C or MPW C

► **NEW! MEMCHECK 2.1 / ANSI \$199**
Includes source code! For any UNIX, VAX, or any ANSI C or K & R projects

SPECIAL BUNDLES!

DOS MASTERS PACK \$199 SAVE \$80!
for Microsoft C and Borland C
WINDOWS GURU PACK \$159
for Microsoft C and Borland C
MICROSOFT POWER PACK \$199
for Microsoft C under DOS and Windows
BORLAND POWER PACK \$199
for Borland C/C++ under DOS and Windows

Order Today!

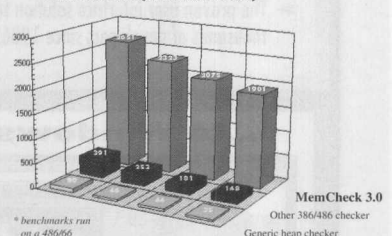
Developing quality software is a **challenge** for even the best programmers. And you have far better things to do than wonder if **the same bugs** will rob you of **productivity, profit, and reputation**. So we invite you to join us as a **valued customer**, and to join the thousands of professional developers all over the world who use MemCheck. Call now **1-800-933-3284 (1-800-WE-DEBUG)** to start **stomping** your worst C bugs! You'll be glad you did—we guarantee it with an **unconditional 30-day money-back guarantee**. One can turn the other cheek only so many times...

MEMCHECK®
ADDS EFFORTLESS RELIABILITY TO YOUR C APPLICATIONS

MemCheck 3.0 for DOS Highlights:

- Detects errors in **third party libraries** without source code or debugging information!
- **Seamless C++** new and delete coverage!
- **Effortlessly pinpoints** memory overwrites, underwrites, memory leakage, invalid frees, null pointer assignments, stack frame overwrites, heap corruption, and other app ills
- Link-only option requires **absolutely no source changes!**
- **Absolutely blazing performance**-- from almost **10 to over 50 times faster** than similar products!
- **Enhanced stack/static/global variable overwrite checking** -- automatically!
- Works **transparently** with any debugger! (TurboDebugger, CodeView, WVIDEO, etc.)
- Can debug problems **at customer sites!**
- **Powerful debugging API**
- **Mouse-driven, user-friendly MemCheck Settings application** for easy control over MemCheck 3.0

SPEED RATINGS (OPERATIONS / SEC)



"MemCheck is worth its weight in gold"

— David Thielen, author of "NO BUGS: Delivering Error-Free Code in C and C++"

FREE BONUS!

VirusCide 3.0 Anti-Virus Software
A \$70 value with every purchase!
(while supplies last)

1-800-WE-DEBUG

StratosWare Corporation

1756 Plymouth Road, Suite 1500 • Ann Arbor, MI •
48105-1890 • **1-800-WE-DEBUG** • International
(313) 996-2944 • Fax Lines (313) 996-2955 • (313)
747-8519 • CompuServe 70244,1372

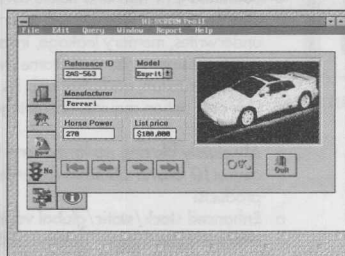
WE USE AND SHIP QUALITY
RECYCLED MATERIALS.

MemCheck is compatible with all linkers, overlay managers, IDE's, memory managers, extenders, and most other debuggers. Overnight delivery available. Call StratosWare for more information.

◆ Request 284 on Reader Service Card ◆

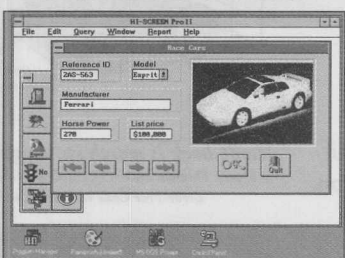
For:
C
C++
Basic
Fortran
Pascal
Cobol
Clipper
dBASE

From DOS...



- Complete graphical interfaces for your DOS programs: data entry screens, windows, help systems, icons, menus...
- Powerful interactive design tools let you create re-usable interface objects in a snap!
- The proven user interface solution for thousands of developers since 1986!

...to Windows



- Just recompile, and see your screens and menus run under Windows. It's that easy!
- Single code base portable across DOS and Windows.

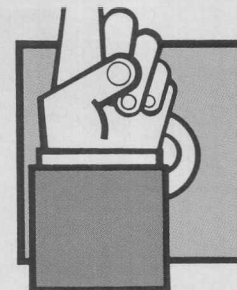
1-800-338-2852

SOFTWAY

Tel: (415) 896-0708 Fax: (415) 896-0709

◆ Request 186 on Reader Service Card ◆

Editor's Forum



Well, I made it past my 50th birthday. Being a 50-year-old computer programmer is definitely better than being 20 — you get more toys to play with and more autonomy in choosing what to do with them. But I confess that it's not as nice as being 30, or even 40 — I miss the energy that came with those relatively youthful milestones. My main consolation is that I still have some energy left, and I'm much wiser about how I spend it now than in years gone by.

I couldn't help but notice at 30 that many of my 40-year-old colleagues weren't as active in the trenches as I was. Ten years later, I found out why. With maturity in one's profession comes increasing demands to serve as caretaker rather than front-line contributor. Someone has to write all those proposals, job descriptions, requirements documents, requisitions, etc. They eat time like candy and sure don't resemble programming. But only the most dedicated techies can resist the siren lure of responsibility. The rest of us get suckered into acting like grownups.

I couldn't help but notice at 40 that many of my 50-year-old colleagues weren't even fighting interesting battles, or so it appeared to me at the time. They seemed to attend interminable meetings and talk about policy matters and other ephemeral abstractions. I mean, how important can it be to draft international standards, or procedures for software quality control, for heaven's sake? Now, after a decade or more of doing that sort of stuff, I've come to see the merit in it. I've learned how to play the guru, or the statesman, or the doddering old fool, as the need arises. I can even sit through a two-hour meeting without squirming (excessively).

I still write the odd bit of code, or the odd requirements spec, and it's more fun than ever. I spent the half-week before my birthday in New Jersey, beating on the draft C++ standard with Andy Koenig, Tom Plum, Bjarne Stroustrup, and others — and I have to admit it was mostly enjoyable. The computer business has never been more exciting than it is today. I have much to be grateful for.

When I stumbled into this field at the age of 19, I never dreamed it would consume my entire adult career. Or that it would bring me so many rewards. I can only hope that most of you who read this magazine can enjoy a comparable passion. May your candle burn equally bright.

So how does it feel to be half a century old? My favorite quote on that topic is from Lowell Thomas, who was asked on his birthday how it felt to be 80. He said, "It's not bad, considering the alternative."

P.J. Plauger
 pjp@plauger.com

Tools.h++ Version 6.0 Now Internationalized!

**Collection Classes
Plus Much More!**

Tools.h++, the best selling industry standard C++ library, now includes Internationalization! A complete, efficient and versatile toolbox of over 100 C++ classes Tools.h++ will make virtually any programming job easier.

And now new Version 6.0 includes:

Internationalization

- Multi-byte and wide character strings
- Localized string collation
- Parse and format times, dates, and currency in multiple locales
- Support for multiple time zones and daylight savings rules
- Support for localized messages
- Localized I/O streams
- Many locales can be active simultaneously

Multi Thread safe

- Safe for use in multi thread environments
- Support for task specific data

Exception

- Comprehensive exception hierarchy
- Exception handlers can report in the local language

Comprehensive test suites are now available!

- ToolsPro.h++ includes a complete test suite for all classes

*Now you can create one executable and ship it to multiple countries!
Tools.h++ lets you read and print times, dates, numbers, and currency in the local format and language!*

*Includes template and non-template classes!
You choose!*

**Now also available on
WINDOWS NT and Macintosh**



Tools.h++ is a complete, efficient and versatile toolbox of over 100 C++ classes:

- String and Character manipulation classes.
- Singly and Doubly linked lists, Stacks, Queues and Vectors classes.
- Smalltalk™-like Collection classes: Set, Bag, SortedCollection, Ordered Collection, Dictionary, Stack Queue, etc.
- Regular Expression Class for search and replace.
- Tokenizer Class for easy string parsing.
- File Class to handle file manipulation with read, write, seek, erase, etc.
- B-tree Class to handle efficient keyed access of disk records.
- File Space Manager Class to allocate, deallocate and coalesce space within files.

- Virtual and Buffered Page Heap to manage objects bigger than 64k.
- All objects fully persistent.

Rogue Wave's Tools.h++ is an "industrial strength" library used in many commercial applications, small and large. All classes have not been derived from a single root object, so they can be easily integrated with other class libraries.

Tools.h++ is an excellent example of how to write true C++ code correctly. Source Code is included. All non-Windows classes are strictly portable between DOS and UNIX.

Other Rogue Wave Libraries Include:

View.h++

A complete C++ encapsulation of the industry standard OSF/Motif tool kit.

LAPACK.h++ and Math.h++

High level math libraries.

DB.h++

A portable C++ database interface for RDBMS's.

Our libraries work with most of the C++ compilers on the market today.



TO ORDER CALL 1-800-487-3217

P.O. Box 2328 • Corvallis, Oregon 97339 • 503-754-3010 • 800-487-3217 • FAX 503-757-6650

◆ Request 124 on Reader Service Card ◆

C++ Language Support Library

Introduction

I conclude my discussion of the language support portion of the library specified by the draft C++ standard. (See "Standard C: The Header `<exception>`," *CUJ*, February 1994, "The C Library in C++," *CUJ*, December 1993, "C++ Library Ground Rules," *CUJ*, November 1993, and "Developing the Standard C++ Library," *CUJ*, October 1993.)

"Language support" consists of those functions that can be called implicitly by C++ code, even when the code apparently contains no function calls. It also consists of the types required to declare and use those functions, as well as a few other related functions and types not directly needed to support the C++ language proper.

Standard C has few such creatures (if any). You can argue that several type definitions are part of language support. The types `ptrdiff_t`, `size_t`, and `wchar_t` are defined in various headers so you can declare objects that have the same types as certain expressions. They are a way to convey otherwise unknowable (or hard to learn) information about these types from the translator to the program.

You can also argue that function `exit` is part of language support. The execution of any C program effectively occurs by evaluating the expression `exit(main(argc, argv))`. Saying such a thing simplifies descriptions — it ties together the effects of calling `exit` and returning from `main`, for example. But it doesn't have a dramatic effect on how you actually write C programs. You cannot, for example, provide your own version of `exit` and expect it to be called when `main` returns. (And many implementations don't really call `exit` when `main` returns, but some other underlying function instead.)

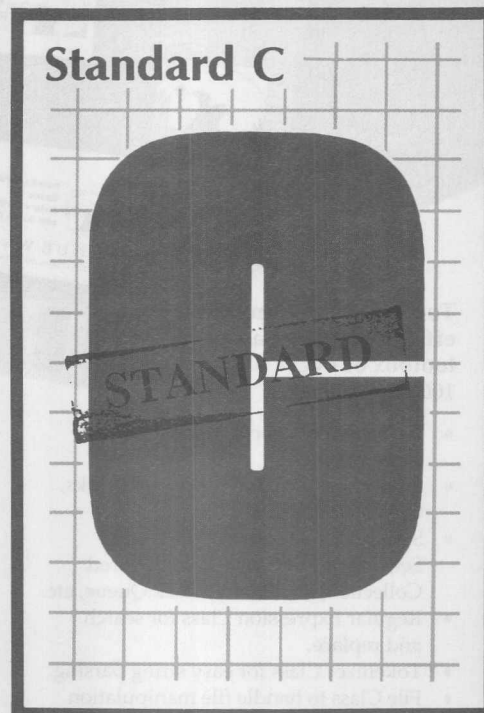
C++, on the other hand, offers numerous opportunities along these lines. You can trot up all sorts of functions that get control, either directly or indirectly, when one of the language support functions gets called. For example, in last month's discussion of exceptions I identified three functions that let you register *handlers*, or functions that get control under some circumstances:

- `set_terminate`, to specify a handler for calls to `terminate()`
- `set_unexpected`, to specify a handler for calls to `unexpected()`
- `xmsg::set_raise_handler`, to specify a handler for calls to `xmsg::raise()`

You can also derive a class from `xmsg` and override the virtual `xmsg::do_raise` to get control when certain exceptions get reported by executing code.

Thus, the draft C++ standard is a bit harder to write than the C Standard. In C, the implementation provides all library functions and you the programmer cannot displace them. The C Standard only has to describe a single interface between implementation and program. In C++, however, the program can displace functions otherwise supplied by the library. The draft C++ standard must spell out the environment promised to such a displacing function. And it must spell out what is expected of the displacing function so the program doesn't get surprised.

A handler for `terminate()`, for example, is not supposed to return to its caller. If you provide one that prints a message and returns, you can cause the library severe problems. The draft C++ standard says so. So when you read the descriptions that follow, remember that the "treaty" between programmer and implementor can be multifaceted. The extra complexity of the draft C++ standard is one of the prices we pay for extra flexibility in this area.



P.J. Plauger is senior editor of The C Users Journal. He is convenor of the ISO C standards committee, WG14, and active on the C++ committee, WG21. His latest books are The Standard C Library, and Programming on Purpose (three volumes), all published by Prentice-Hall. You can reach him at pjp@plauger.com.



New Borland C++ 4.0 Visual is just the beginning

Why settle for ordinary visual when new Borland C++ gives you so much more? The highest quality fourth-generation tool set. A fully customizable and open desktop. The ability to target

16- and 32-bit Windows simultaneously. And, of course, it's "visual."

Only Borland C++ gives you a fully integrated professional editor featuring BRIEF® technology, powerful Turbo Debugger® GX, and integrated C and C++ VBX control support.

An environment to die for

Borland C++ 4.0 takes the bureaucracy out of development. With the visual Project Manager and its multi-target capabilities, even the most complex projects are handled automatically. The flexibility of AppExpert actually

generates much of your application for you. TargetExpert, ClassExpert™, DialogExpert, and Resource Workshop™ streamline the tasks of setting up and customizing your applications.

A true C++ implementation that's years ahead

Languages evolve for a good reason—so programmers can realize ever-increasing productivity and safety of code. That's why Borland is the first to bring important new enhancements to the C++ language, like full support for templates, exception handling, and Run-time Type Information. And Borland C++ includes ObjectWindows™ Library (OWL) 2.0—the world's most popular framework.

Borland C++ is the world-standard C++ because it's the easiest to use, yet has power to spare for the most complex tasks. If you're serious about C++ programming, new Borland C++ 4.0 is the environment you've got to

have. After all, if your compiler is only visual, it's just a facade. Get new Borland C++ 4.0 for Windows and DOS now.

Borland C++ owners
Upgrade Now!

\$149⁹⁵
(Suggested list price \$499.95)

Other C++ owners
\$199^{95*}

90-day, money-back guarantee!

See your dealer or call now,

1-800-336-6464, ext. 7086

In Canada call, 1-800-461-3327

Borland
Power made easy™

Copyright © 1993 Borland International, Inc. All rights reserved. All Borland product names are trademarks of Borland International, Inc. *Offer good for owners of Microsoft and Symantec C or C++ products; also Turbo C++ owners who purchased product before November 1, 1993. Offer good in the United States and Canada only. All prices in U.S. dollars. Dealer prices may vary. BI 5827

◆ Request 465 on Reader Service Card ◆

Standard C programming. Similarly, the addition of *new* and *delete* to C++ essentially structure the use of *malloc* and *free*. By writing:

```
Thing *p = new Thing;
```

you are assured that the object of type *Thing* is properly constructed after it is successfully allocated and before it can be accessed through *p*. Similarly, the expression statement:

```
delete p;
```

ensures that the object is destroyed before its storage is deallocated.

You don't have to include any headers before writing expressions like these — *new* and *delete* are indeed built right into the language. But you can also play a variety of games with storage allocation if you choose. To do so, you begin by including the header `<new>`. Listing 1 shows a representative version of this header. I omit the extra superstructure required by namespaces, because it is distracting and still in a state of flux.

The simplest game you can play is to gain control when space for the heap is exhausted. The function *set_new_handler* lets you register a handler for this condition. In principle, the draft C++ standard says you can "make more storage available for allocation and then return," but it fails to describe a portable way to do so.

One or more allocated objects may also help, but even that is not guaranteed. More likely, you will want to throw an exception or terminate execution at this point.

xalloc Exceptions

The default "new handler" does, in fact, throw an exception now. As I described last month, all library exceptions are derived from the base class *xmsg*. Moreover, all exceptions are thrown by calling *ex.raise()*, for some object *ex* of class *xmsg*. Unless you seize control of the process in one of the ways I described last month, the eventual outcome is that a failed allocation will throw an exception, which will in turn terminate execution of the program.

This is a significant change from universal past practice, which has been to quietly yield a null pointer as a result of the *new* expression. The Library Working Group of X3J16/WG21, the joint ANSI/ISO standards committee for C++, anguished quite a bit before recommending this change. The joint committee anguished a bit more in turn. But eventually, the predominant wisdom was that the Standard C++ library had bloody well better use the full language in this case, not just the bits that were available when *new* and *delete* were first added to C++.

A persuasive argument is that very few programs truly check *all new* expressions for null pointers. Those that don't may well stumble about when the heap is exhausted — they're almost certainly better off dying a clean death. Those that do check all such expressions often simply abort — the path to abnormal termination is now just slightly different. It is only those few sophisticated programs that try to do something nontrivial when heap is exhausted that need a bit of rewriting. Most of the joint committee felt this was a necessary price to pay to introduce exceptions at this critical juncture.

Even so, some sympathy remains for being able to revert to the old behavior. For a variety of reasons, the Library Working Group has not spelled out a portable way to do so. But the group has

**Publication Quality
GraphiC
Scientific Graphics**

Orbits correspond to $J=\text{constant}$ contours

$J = 1 - (r^2 - 1)^2 + \epsilon \cos(\theta)$

Version 7.0 For DOS, DOS '286, OS/2, Windows and Windows NT

- Four-times higher resolution
- 248 colors for all plot elements
- Log plots to any base
- 3-D surfaces shaded to level contours
- Printer output limited only by size of output medium (DOS)
- New — patch plots, box and whisker plots, staircase plots

- Convert high-resolution output to PIC, GEM, HPGL, HPGL2, CGM, SCODL, TIFF, PostScript (Level 1 and 2), & Tektronix 4105 formats
- Create color separations
- Statistical and smoothing functions
- Use any of your PostScript or True-type fonts in GraphiC for only \$59

GraphiC is a C-library that allows you to create every sort of scientific and engineering plot. No special knowledge is required to program in GUI environments.

Scientific Endeavors Corporation
 508 N. Kentucky St., Kingston, TN 37763
 (800) 998-1571 FAX: (615) 376-1571

Listing 1 The header `<new>`

```
#ifndef _NEW_
#define _NEW_
#include <exception>

// class xalloc
class xalloc : public xruntime {
protected:
// virtual void do_raise();
public:
xalloc(const char * = 0, const char * = 0);
virtual ~xalloc();
};

// function and object declarations
fvoid_t *set_new_handler(fvoid_t *);
void operator delete(void *);
void operator delete[](void *);
void *operator new(size_t);
void *operator new[](size_t);
void *operator new(size_t, void *);
extern fvoid_t (*_New_hand);
#endif
```

☐ Request 172 on Reader Service Card ☐

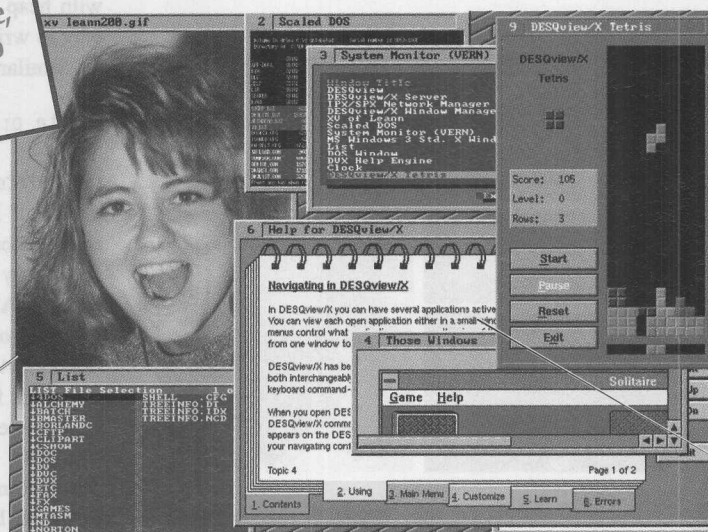
The simplest way to integrate DOS, MS Windows and X Window software on your PC—or any other PC on a network.

New Version 1.2 includes enhanced network support: PC-NFS, Hewlett-Packard LAN Manager, Microsoft LAN Manager, Beame & Whiteside, Wollongong—in addition to PC/TCP and Novell LAN Workplace for DOS

DOS and Windows applications compatibility.

Proven memory management and diagnostics with both Manifest and QEMM.

DESQview/X has EGA, VGA, 8514A, Super VGA and DGIS high resolution graphics support.



Comfortable 'look and feel' for both keyboard and mouse users.

With scalable window capability provided by Adobe Type Manager™, you can get more DOS windows on the screen than ever before.

Any DESQview/X window can be made into an icon at any time.

Online help is two keystrokes away.

Multitasking Power

DESQview is the recognized pioneer in DOS multitasking and is at the very heart of DESQview/X—giving you proven high performance multitasking of DOS and Microsoft Windows programs side-by-side.

X Window System Graphics

The X Window System gives DESQview/X its graphical interface. And its ability to access DOS text and MS Windows programs running remotely on your network. Add the optional TCP/IP Network Manager and it gives you the ability to access remote X Window programs on workstations such as an IBM RS/6000 or a SUN SPARCstation. Or vice versa.

Adobe Type Manager

The accepted standard in scalable font technology is built in so that X Window System programs can use scalable fonts. And it also gives you the ability to view DOS text programs in scaled windows.

Program Memory Management

DESQview/X has the memory management services of advanced operating systems—thanks to the built-in Rational Systems DOS4GX DOS extender (16 and 32 bit), shared libraries, dynamic link libraries (DLLs) and virtual memory.

Superior DOS Memory Management

DESQview/X comes with our award-winning, best-selling expanded memory manager, QEMM, and the highly acclaimed Quarterdeck Manifest to ensure that your PC's memory is always at its optimum and that your DOS programs have the absolute maximum memory available.

Quarterdeck Office Systems, 150 Pico Boulevard, Santa Monica, CA 90405 (310) 392-9851 Fax (310) 314-4219
Quarterdeck International Ltd., B.I.M. House, Crofton Terrace, Dun Laoghaire Co. Dublin, Ireland Tel.(353) (1) 284-1444 Fax: (353) (1) 284-4380

©1993 Quarterdeck Office Systems. Trademarks are property of their respective owners.

◆ Request 201 on Reader Service Card ◆

Customization

With DESQview/X's customizable menus, graphical desktop, keystroke macros, mark and transfer and online help, it is very easy to tailor the your own PC to your exact working needs. In addition, you can choose from two optional 'look and feel' packages: OSF/Motif and OPEN LOOK.

Includes Valuable Companion Programs

DESQview/X comes with three companion programs. Application Manager for launching and organizing programs. File Manager for managing local and remote files. And Icon Editor for creating icons.

Quarterdeck makes your PC a better place to work, whatever software you prefer to use.

Qty	Product	5-1/4	3-1/2	Each	Totals
	DESQview/X			\$275.00	
	DESQview/X Network Mgr. (TCP/IP)			\$200.00	
	DESQview (for 286 PCs)			\$99.95	
	DESQview 386 (for 386 and 486 PCs)			\$149.95	
Shipping & Handling \$10 (Canada & USA only)					
California Residents add 8.25%					
Grand Total					

Call (800) 354-3222 ext. 1D1 for fastest service
Please allow 3 weeks for delivery

Yes! Please Send: ☐ Visa ☐ MasterCard Expires ____/____/____ Card # _____

Payment method: _____

Name _____ Address _____ City _____ State _____ Zip _____

Quarterdeck

identified what it thinks should be a common extension. Calling `set_new_handler` with a null pointer argument is otherwise undefined behavior. It seems natural to use this nonportable call as a way for implementations to know that they should revert to the older behavior.

Replacing `operator new(size_t)`

If you want more certain control over the business of allocating storage, your best bet is to provide your own versions of `operator new(size_t)` and/or `operator delete(void *)`. These functions have a peculiar dispensation — the library provides a version of each, but you can “knock out” those versions by defin-

ing your own. (Only the array versions of these two operators, described below, also enjoy this special status within the Standard C++ library.)

Before I go into details, please note an important distinction here. When you write:

```
Thing *p = new Thing;
```

the `new Thing` part is called a “*new expression*.” It calls `operator new(size_t)` to allocate storage, but it also does other things, such as constructing the newly allocated object. All that `operator new(size_t)` has to worry about is providing the number of requested bytes, suitably aligned, or dealing with heap exhaustion. Listing 3 shows one way to write this function.

Similarly, when you write:

```
delete p;
```

the `delete p` part is called a “*delete expression*.” It calls `operator delete(void *)` to free storage, but it first destroys the object (only if the pointer is not null, of course). All that `operator delete(void *)` has to worry about is freeing storage for the object. Listing 4 shows one way to write this function.

So one thing you might do is replace `operator delete(void *)` with a function that doesn't really free the storage. That could be handy while you're debugging a program, provided of course that you have enough heap to run your test cases.

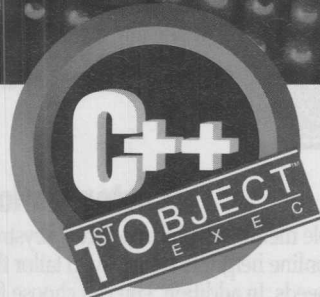
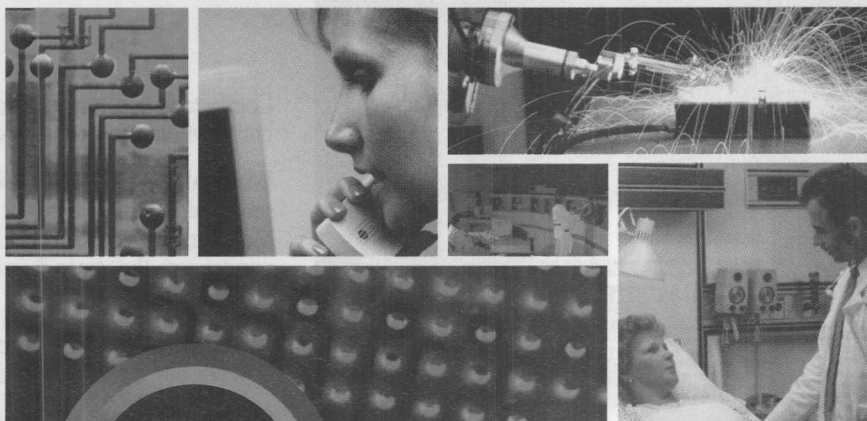
Or you might replace both `operator new(size_t)` and `operator delete(void *)` with versions that are simpler, or faster, or more sophisticated than the library versions. It is important to replace both, because the latter function in the library only knows how to free storage for objects allocated by the former.

In either case, you probably don't have to bother with `set_new_handler`. You are at liberty to do whatever you want when you run out of heap. No need to call the new handler, which you can't easily do portably anyway.

Placement Syntax

Yet another latitude granted by the C++ language is to provide an arbitrary set of additional arguments in a *new expression*, as in:

```
Thing *get_special(T1 stuff,
                  T2 more_stuff)
{
```



True Object-oriented C++ Multitasking Executive for Real-time, Real World Applications

The hub of any embedded or native real-time C++ application should be a **true** object-oriented, multitasking executive—like 1stOBJECT EXEC. It enhances your productivity and gives you all the advantages that complete C++ offers—access to reusable software faster.

1stOBJECT EXEC from DDC-I (a global leader in Ada compiler technology) is object-oriented, fully developed from scratch in C++ using the object oriented paradigm.

It lets you implement real-time designs in C++ compatibly, without the problems associated with using a C-based kernel.

Designed for the needs of real world products, a family of 1stOBJECT EXEC versions for embedded targets will help you reduce development cost and time—and reuse code for future needs.

Typical applications? Medical diagnostics, communication systems, industrial robotics, intelligent terminals, test equipment or peripheral controllers.

Call for details and host/target availability.

- Sun™ SPARC® → 680x0 available now
- Sun™ SPARC® → 683xx 30 days ARO
- PC/DOS™ → 680x0 available now
- PC/DOS™ → 683xx 30 days ARO
- PC/DOS™ → 80x86 available now



410 N. 44th Street, Phoenix, AZ 85008, (602) 275-7172 FAX (602) 275-7502
Gl. Lundtoftevej 1B, DK-2800 Lyngby, Denmark, +45 45 87 11 44 FAX +45 45 87 22 17

◆ Request 218 on Reader Service Card ◆

Prepare Yourself for Visual Reality.

Introducing ProtoGen+. Visual tools with the most awesome workbench ever created for Windows development.

The future has arrived—a complete point-and-click, WYSIWYG workbench that lets you create dazzling applications without writing a line of code.

Discover the ease and productivity of visual development!

Visually develop screens

Create a menu and connect screens & dialogs

Test the application's screen flow in a live environment

Instantly generate source code in Pascal, C or C++

Paint your screens. Design a menu and link screens together. Test the flow in a live environment, and generate code for ANSI C, C++ for OWL or MFC or Pascal with objects. It's that easy.

And this open! ProtoGen+ will work with your database, compiler,

libraries and extensions. Powerful add-on features, like SQLView offer

workbench access to multiple databases to develop client/server and xBase applications. Snap-in components make ProtoGen+ open to future development technologies—whatever they may be.

ProtoGen+ is easy to learn, speeds your development cycle and protects your investment by generating C, C++ and Pascal. We guarantee you'll prototype and generate exciting applications faster than you ever thought possible!

Point-and-click to paint screens with bitmaps, icons, tables, data validation, custom colors, fonts, 3D effects, visual toolbars, status lines, balloon help, MDI and more!



PROTOVIEW™
The Visual Development Edge™

All products named are trademarks of their respective companies. ©1993 ProtoView Development

◆ Request 376 on Reader Service Card ◆

The most powerful, open set of Visual Development Tools ever!

Build windows, forms, dialog boxes, tool bars

Dialog Editor

Menu Designer

Quickly create a menu using templates

Output C, C++ and Pascal with Objects

Code Generator

ProtoView Screen Manager

Data validation, 3-D effects, MDI and more

Create new designers! Source included

Custom Visual Designer

Win-Control Library

Rich library of visual control objects

Snap-in Code Generators

License our technology to create new code generators

Fasten your seatbelt for ProtoGen+!

Only **\$199** (list price \$395)

1-800-231-8588

Ask for Ext. 40

In NJ, call (908) 329-8588

ProtoGen+'s SQLView access to multiple databases is available at an additional price; ask about it when you call.

Bring your applications to life using the latest visual design tools!




```
return (new (stuff, more_stuff) Thing);
}
```

This form implicitly calls the function:

```
void *operator new(size_t, T1, T2);
```

which you are obliged to supply. I leave it to your imagination what extra parameters might be useful when you're allocating some of your more sophisticated objects.

It doesn't take too much imagination, however, to see a very common need. Sometimes you know exactly where you want a C++ object to be constructed — you have reason to believe that the storage area *X* is large enough and suitably aligned to hold an object of type *Thing*. Moreover, you're confident that no object has been constructed there already for which a destructor will later be called. (Whew!)

To deal with this twilight zone between C and C++ programming, you can write:

```
Thing *p = new ((void *)&X) Thing;
```

This, naturally enough, calls the function:

```
void *operator new(size_t, void *);
```

which can simply return its second argument, as shown in Listing 4. The Standard C library provides this one version of a placement operator *new*. (Don't forget to include the header *<new>* to be sure it is properly declared.) Any fancier placement variants are up to you to provide.

Member operator new

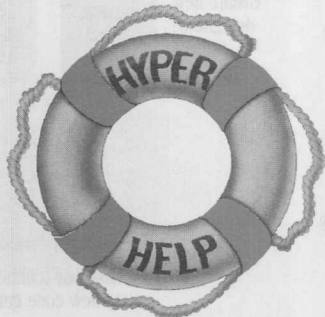
Yet another way exists for controlling how objects get allocated. For any class, you can overload all the variants of *operator new* and/or *operator delete* that I've mentioned so far. Perhaps you want to write your own versions of:

```
void *Thing::operator new(size_t);
void Thing::operator delete(void *);
```

that does a really fast job of allocating and freeing objects of class *Thing*. It can, for example, maintain a list of previously freed objects and hand them back quickly for future allocation requests. Unless you really get tricky, you can even ignore the *size_t* first argument to all variants of *operator new*, since you know how big a *Thing* is likely to be. (How do you get tricky? Well, you can make *operator new* virtual in the base class and fail to override it in a derived class. But thinking about things like that gives me a headache.)

So you see that you can exercise pretty fine control over how all objects, or even individual objects, get allocated.

Offer the HyperHelp Advantage



and be in good company!

WordPerfect • AT&T
Computer Associates (20/20) • Sybase
Siemens • EDS • Swiss Bank
Merrill Lynch • Corel (CorelDRAW!)
SPSS • Lotus (Ami Pro)

HyperHelp™ - The Industry Standard,
On-line Help Facility for Motif

Bristol Technology Inc.

Ph: (203) 438-6969 Fax: (203) 438-5013 email: info@bristol.com
FTP a demo: bristol.com, demo directory

Listing 2 The function operator new(size_t)

```
// operator new(size_t) REPLACEABLE function
#include <stdlib.h>
#include <new>

void *operator new(size_t size)
{
    // try to allocate size bytes
    void *p;
    while ((p = malloc(size)) == 0 && _New_hand != 0)
        (*_New_hand)();
    return (p);
}

// End of File
```

Listing 3 The function operator delete(void *)

```
// operator delete(void *) REPLACEABLE function
#include <stdlib.h>
#include <new>

void operator delete(void *p)
{
    // free an allocated object
    free(p);
}

// End of File
```

◆ Request 259 on Reader Service Card ◆

cation and freeing of arrays. You can, for example, write:

```
Thing *p = new Thing[N];
```

to allocate an array of *N* elements each of type *Thing*. Each of the elements is constructed in order, starting with the first (element zero). In this case, you *must* write the expression statement:

```
delete[] p;
```

to delete the array, not just a simple:

```
delete p;
```

as before. Why? Because the "array *new* expression" above has to somehow memorize how many elements *N* it has allocated. It needs to know to locate this memorized information and use it to destroy the appropriate number of elements and free the appropriate amount of storage. Yes, some existing implementations of C++ let you be cavalier about deleting arrays the wrong way, but don't count on that license in a portable program.

This requirement presents another problem. What happens if you've provided a member *operator new(size_t)* for class *Thing*, as above? It cannot, in general, know whether it's being asked to allocate storage for a single element or a whole array. (Remember the potential trickery I mentioned above.) So what C++ has done in the past is to ignore any such member functions and call the global *operator new(size_t)* for all array allocations. This has been a less than satisfactory solution.

The joint committee has plugged this control gap by permitting you to define functions such as the members *operator[] new(size_t)* and *operator delete(void *)*. Defining these functions gives you control over the allocation and freeing of

called, by the way. The array *new* expression can ask for stack storage for its own bookkeeping, so you'd better honor the *size_t* argument blindly. But at least you can maintain private storage pools now for array objects.

For completeness, the draft C++ standard also includes global versions of:

```
void *operator new(size_t);  
void operator delete(void *);
```

Windows Controls for C / Pascal

Extensive Layout Settings

- Any text font and size (ATM, TT, System).
- Text and background colors freely selectable.
- Unique Transparent Mode for static text fields.
- Vivid 3D Appearance.
- 3D depth and height freely adjustable.
- Animated displays in buttons.
- User-defined bitmaps in all controls.

With the integrated Kolibri Resource Manager you can also develop extensive projects very quickly, and save on Windows System Resources.

Support comes free direct from the developers by telephone, fax, CompuServe® and Mailbox.

YOU GET 14 CONTROLS + A DETAILED HANDBOOK + WINDOWS HELP FILES + EXTENSIVE DEMO SOURCE CODE PROGRAMS FOR ONLY \$349

KOLIBRI

Custom Control Library

NORTH & SOUTH AMERICA
European Software Connection
P.O. BOX 1982, Lawrence, KS 66044, USA
phone: (913) 832-2070, fax: (913) 832-8787
CompuServe 71141,3624

ELSEWHERE CONTACT
Softronic GmbH, Englerstr. 9
D-77652 Offenburg, Germany
phone: (+49) 781 74021, fax: (+49) 781 74023
CompuServe 100031,510

Get your free demo today!

WINDOWS is a trademark of Microsoft Corporation

With the unique facilities offered by the extensive Kolibri Control Library you can considerably reduce your development costs and give your program a polished professional appearance.

Kolibri simply plugs into Resource Workshop. All controls are designed through clear and easy to use dialog boxes. And, of course, controls can also be temporarily generated by a program and driven by messages.



Listing 4 The function operator new(size_t, void *)

```
// operator new(size_t, void *)  
#include <new>  
  
void *operator new(size_t, void *p)  
{ // allocate in place  
  return (p);  
}  
  
// End of File
```


The library versions of these functions just turn around and call the non-array library versions, so I won't show you the code for them. And you can indeed knock these functions out with your own definitions, but I'm not sure why you'd bother. Doubtless, someone more clever or perverse than I can make a case for *any* feature added to C++.

Type Information

There is one last aspect to the language support library. It is rather small compared to exceptions (all of last month's installment) or storage management (most of this month's). I tack it on here for completeness.

Another relatively recent significant addition to the draft C++ standard is "run-time type identification" (or RTTI, for short). Basically, it adds the operator *typeid* for obtaining various bits of information on the type of an object (or expression). The operator yields an object of class *typeinfo*, defined in the header *<typeinfo>*. Listing 5 shows one way to write this header.

The exception *badtypeid* is reported in those cases where the type cannot be determined statically at translation time. If, in the process of chasing down the actual object, the program encounters a null pointer, you can guess what happens.

(If you're put off by all these names made from words run together, you're not alone. There's a good chance that the joint committee will approve a new naming convention that involves a more liberal use of underscores to separate component words in

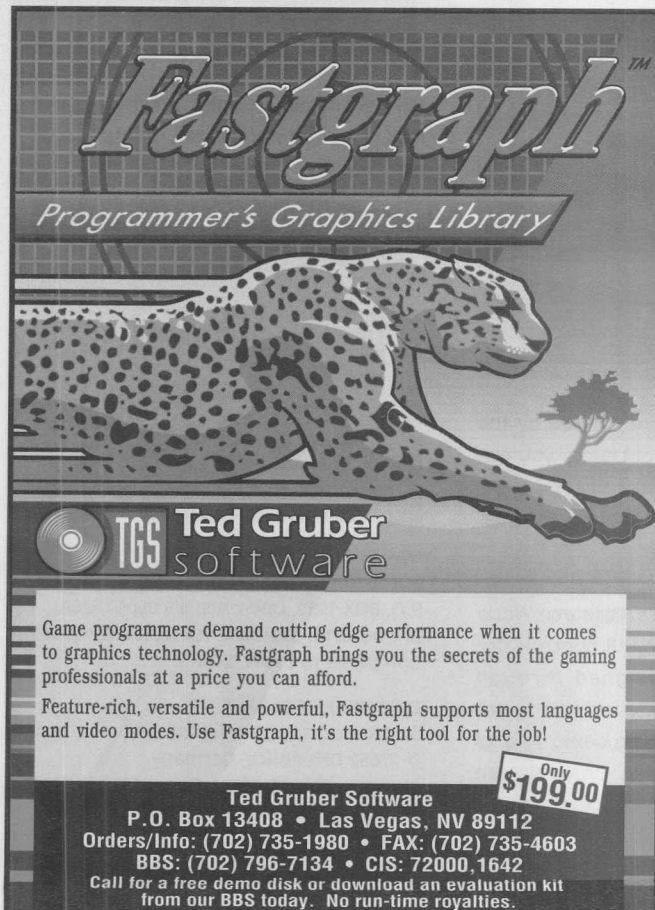
names. So don't be surprised if many of these compound names change in the coming months.)

What can you do with an object of class *typeinfo*? Well, you can obtain some sort of name for the type, for one thing. *typeinfo::name()* yields a null-terminated multibyte string (or NTMBS, in the jargon of the draft C++ standard) that presumably says something meaningful about the type. There are no standard names defined, so far, not even for the builtin types.

You can also compare two objects of class *typeinfo* for equality or inequality. Within any given program, you can expect two such objects to compare equal only if they derive from two expressions of the same type. Don't expect to be able to remember these critters in files, however, and check for type equality across programs. Even running the same program twice doesn't promise to yield the same representation of a *typeinfo* object for the same type each time. (I have indicated that the type information can be represented as an *int*, but that is just illustrative, not a requirement.)

Finally, you can impose an ordering on all the types within a program. *typeinfo::before(const typeinfo&)* returns nonzero for an object that represents a type earlier in the pecking order than the argument object. Once again, however, no promises are made about the rules for determining this order, or whether they're even the same each time you run the program.

I'm sure far more can be said about the uses of RTTI, but I'm not the one to say it at this point in my career. Even if I were, this is not the place to say it. For now, you know what the standard C++ library has to know about RTTI. □



Fastgraph™
Programmer's Graphics Library

TGS Ted Gruber software

Game programmers demand cutting edge performance when it comes to graphics technology. Fastgraph brings you the secrets of the gaming professionals at a price you can afford.

Feature-rich, versatile and powerful, Fastgraph supports most languages and video modes. Use Fastgraph, it's the right tool for the job!

Ted Gruber Software
P.O. Box 13408 • Las Vegas, NV 89112
Orders/Info: (702) 735-1980 • FAX: (702) 735-4603
BBS: (702) 796-7134 • CIS: 72000,1642
Call for a free demo disk or download an evaluation kit from our BBS today. No run-time royalties.

Only \$199.00

Listing 5 The header *<typeinfo>*

```
#ifndef _TYPEINFO_
#define _TYPEINFO_
    // class badtypeid
    class badtypeid : public xlogic {
    protected:
    //     virtual void do_raise();
    public:
        badtypeid();
        virtual ~badtypeid();
    };

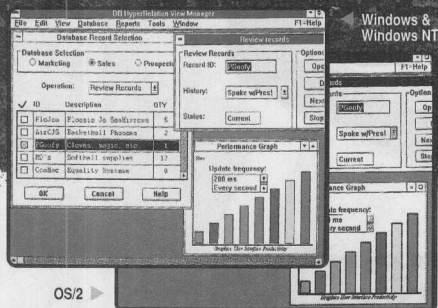
    // class typeinfo
    class typeinfo {
    const char *_Name;
    const int _Desc;          // implementation dependent
    typeinfo(const typeinfo&);
    typeinfo& operator=(const typeinfo&);
    public:
        virtual ~typeinfo();
        int operator==(const typeinfo&) const;
        int operator!=(const typeinfo& _Rop) const
        {return !(*this == _Rop); }
        int before(const typeinfo&);
        const char *name() const {return (_Name); }
    };
#endif

// End of File
```

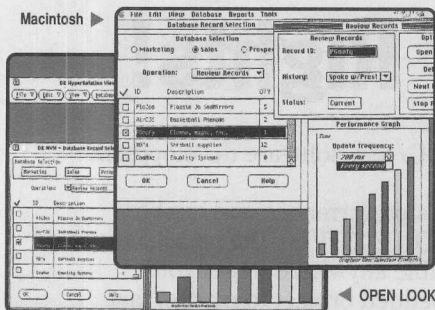
◆ Request 174 on Reader Service Card ◆

WINDOWS WINDOWS NT OS/2 MACINTOSH OPEN LOOK OSF/MOTIF CHARACTER SYSTEMS

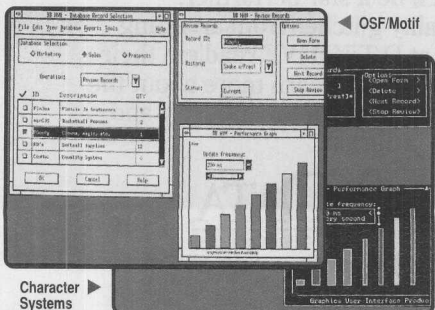
Develop Cross-Platform GUI Applications in a Single Stroke.



OS/2 ▶



Macintosh ▶



OSF/Motif ▶

Character Systems ▶

Master the art of multi-platform GUIs.

XVT Software is the leading choice of world-class developers for one reason: It is the simplest, quickest path to building quality applications that port to every GUI without compromises in look-and-feel or performance. Plus, it's easier to learn and use than native toolkits, so your time and effort goes into your application, not your GUIs.

XVT gives you simultaneous original GUIs.

Because XVT uses native GUI objects, your application is indistinguishable from one written directly to the native toolkit. Through our layered architecture, you achieve equivalent cross-platform functionality appropriate to each GUI, without the overhead and inflexibility of proprietary emulation-based systems.

XVT puts complete C/C++ solutions at your fingertips.

XVT Development Solutions for C include an Interactive Design Tool. Solutions for C++ include an object-oriented application framework. Both include the XVT Portability Toolkit.

When combined with in-depth consulting, training and support, plus a wide range of Partners products, XVT forms the most comprehensive and advanced solution for developing completely portable GUI applications.

◆ Request 197 on Reader Service Card ◆

Developers judge XVT to be a masterpiece.

XVT is the base document for the IEEE's GUI standardization effort. Our thousands of customers include internal and commercial developers like: Alcoa, Amoco, AT&T, Avis, Ford, General Motors, Grammatik/Reference Software, Kodak, Lockheed, NCR, NEC, NIST, Novell, Rockwell, Siemens, Sony, Southwestern Bell, Tandem, Uniplex, Unisys, US Army and US West.

Call now for a Free XVT Demo and Technical Overview.

1-800-678-7988

NEW!
XVT-PowerObjects

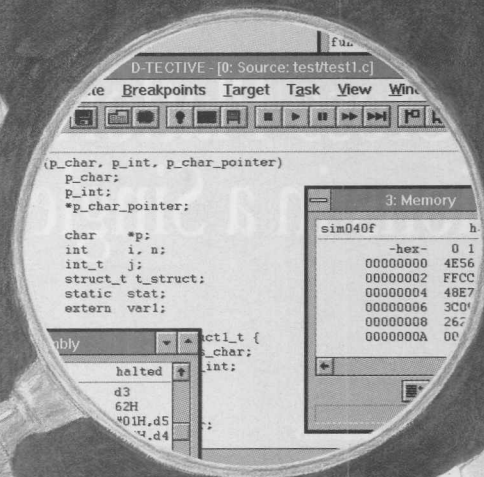


The portable GUI development solution.

XVT Software Inc. 4900 Pearl East Cir. Boulder, CO 80301
(303) 443-4223 FAX (303) 443-0969

For European inquiries, contact: Precision Software GmbH
Phone: 49 0 61 03/37 94 0 Fax: 49 0 61 03/36 95 5

ARRESTED DEVELOPMENT?



Let Diab Data's D-TECTIVE Read Your Bugs Their Rights!

Are your efforts to get production-quality software out the door being dogged by suspicious, recurring, and persistent bugs? Put Diab Data's D-TECTIVE on the beat!

D-TECTIVE is the latest word in GUI, multi-tasking, multi-target, remote, source-level debuggers for Motorola's 680x0 and CPU32(+) family of embedded controllers and microprocessors, and it runs on the most popular platforms, in-circuit emulator tools, and real-time operating systems.

You've already heard about the legendary performance of Diab Data's D-CC/68K—the fastest, most reliable compilers for Motorola's 680x0 family of processors. Now, the industry's recognized leader in high-performance globally optimizing compilers has added proven debugging

technology to its D-CC/68K software development tool arsenal.

Protect your code's performance and quality with D-CC/68K and D-TECTIVE—the smartest one-two punch available anywhere to get your Motorola-based embedded project to market fast and guarantee that it'll be a star performer.

Stop bugs and quality glitches from arresting your development! Let D-TECTIVE read'em their rights! Call Diab Data now to learn more about the most arresting software tools in the industry and our FREE evaluation program.

DIAB DATA
A Bull Company

U.S.: Telephone: (415) 571-1700 • Fax: (415) 571-9068
Europe: Telephone: +46-8 622-4422 • Fax: +46-8 622-4223

Symbolic Access To Embedded Controllers

Odd A. S. Olsen and Petter H. Heyerdahl

Introduction

A PC often functions as the user interface, debugging tool, and general supervisor of an embedded controller. With this arrangement it is important for the programs that transfer parameters between the PC and the controller to maintain agreement on the names and meanings of all variables. If the PC sets parameter 63 to the value of 123, for instance, the PC and the controller must agree on what parameter 63 is and what the value 123 represents. This consistency is especially at risk during development because new parameters are sometimes introduced on-the-fly and often end up in different locations on the two computers. The processors may not even agree on a format for the values. (An Intel PC could be storing numbers in big-endian format while a Motorola controller might want them in small-endian format.)

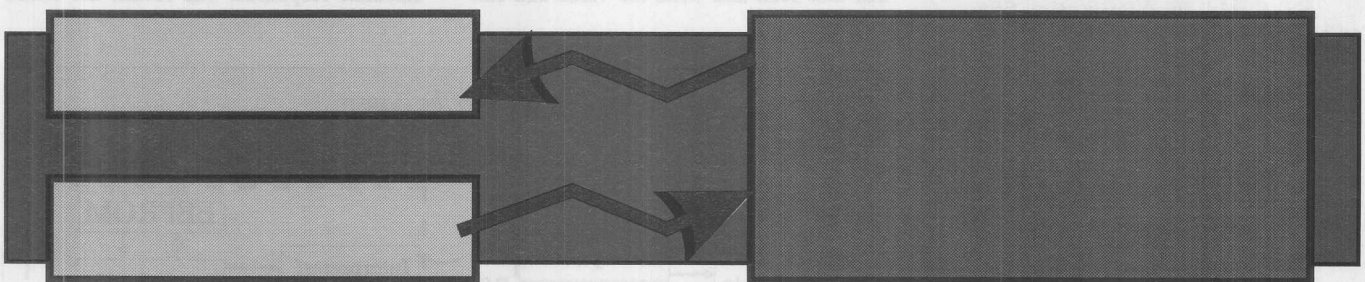
A symbol table is an efficient solution to the consistency problem because it allows the data structures of the PC and controller to develop separately, yet links the two systems with a simple communication protocol. By this method, which resembles the message format found in the general-purpose interface bus (GP-IB), values are referenced by their symbolic name, e.g. *measured_ph*, *heater_power* etc. We used this principle to control a group of bioreactors. Each reactor is controlled by an embedded controller which communicates with a supervising PC for operator instructions and data logging.

How Things are Connected

Figure 1 illustrates the information flow in the system. In our implementation the PC is always the initiator of communication and the embedded controller only responds to received messages. The most important entries in the symbol tables are the parameter names and their values. (In this context a parameter is more general than a constant. It can also represent measured data, debug information, etc.) Parameter values move from the PC symbol table to the controller symbol table through messages sent over the RS-232 line. The PC symbol table can be altered either by accessing the user interface or by causing the PC to read a refreshed version of the parameter file. The new data is then sent to the symbol table in the embedded controller. The controller symbol table can thus be changed from the PC, or through the measurement and control algorithms of the controller itself. Parameters for the control algorithms are stored in the EEPROM, enabling the controller to initialize its symbol table values at start-up and maintain its control process by itself should the PC fail.

The Symbol Tables

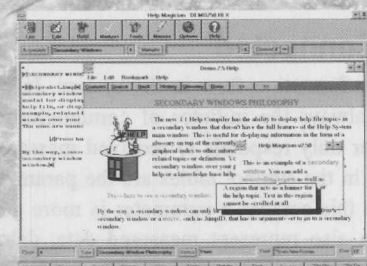
Both the PC program and the embedded controller program are centered on their respective symbol tables. The structure of the PC symbol table is:



Odd A.S. Olsen and Petter H. Heyerdahl received their Master of Engineering degrees in Engineering Cybernetics at the Norwegian Institute of Technology, where Mr. Olsen later received a Ph.D. He is currently an independent consultant, developing electronics and programs for embedded controllers. Mr. Heyerdahl is associate professor at the Agricultural University of Norway, Department of Agricultural Engineering. Odd may be contacted at Jutulveien 11, N-0852 Oslo, Norway or by the Internet: odd.olsen@itf.nlh.no.

Windows Help Magician 2.5

PC Magazine says "Designing help systems with the Help Magician is so easy that you might even want to use it as a hypertext authoring tool, independent of providing help for a product. Certainly for Windows programmers it's indispensable!" - July 1993 Toolkits section



Create, edit, and test help quickly in an integrated environment!

We've sliced the hypertext help development time from days to hours! Forget about footnote or RTF programming or using Word. The Help Magician comes with a 30-day money back guarantee.

New version 2.5 features

- ☐ No memory limitations
- ☐ Supports 3.1 Help Compiler features such as non-scrolling regions and secondary windows
- ☐ Set background colors
- ☐ Advanced paragraph formatting
- ☐ Full macro support
- ☐ Jump to other help files
- ☐ Multiple file support
- ☐ Help window sizing
- ☐ Imports Word & Lotus Ami Pro RTF files
- ☐ Built-in spell checker
- ☐ Multimedia support for audio, video, midi, and animation files
- ☐ Writes #define files
- ☐ Still only \$199!

Order Now! Call
(800) 542-2742

**SOFTWARE
INTERPHASE
INCORPORATED**

82 Cucumber Hill Rd
Suite 227
Foster, RI 02825
Voice: (401) 397-2340
Fax: (401) 397-6814

```
typedef struct {
    char *name;
    int ival;
    double fval;
    void *(*scalefunc)(void *);
    char eeProm;
    char file;} symTabEntry;
```

Each of the process variables maintained by the controller (temperature, oxygen level, pH, etc.) is represented as a record in the symbol table. Every record in the table contains a unique name (e.g. *measured_pH*), an integer or double floating-point value, and a pointer to an optional scaling function. The table also contains two flags: *eeProm* and *file*. If *eeProm* is set to *TO_EE*, the record will be transferred to the controller upon start-up, and will be stored in the controller *EEPROM* if its value is different from the one already residing there. The *file* flag indicates whether the current record contains data associated with the system's parameter file. This parameter file contains parameter names (of process variables and other system variables) and their values. If the system finds a particular parameter name in the file during startup, the system copies the parameter's value into the record, and sets a bit in the *file* flag. Likewise, if the user later wants to save the current parameter values back to the file, the system will save only those parameters whose *file* flags are set. While the system is operating, a record holds a parameter's value in either *ival* (for integer parameters) or *fval* (for floating-point parameters).

The user interface is also oriented to the symbol table. The user can change parameters by choosing a parameter window. Each parameter entry field knows the pointer to its entry in the symbol table. It can thus read and write the value and call

Odd A.S. Olsen and Petter H. Heyerdahl

the scaling function. When the user changes a parameter, the new value is immediately scaled from floating-point to integer and transmitted to the controller. Because values are referred to by name rather than by pointer, the set-up of the parameter windows is simplified.

The symbol tables are alphabetically sorted, which allows the processor to use a binary search when looking for a name. (A binary search is faster than a linear search, but if the tables were large, I would consider using hash-tables.) The programmer will now and then insert new entries to the tables at a wrong position. On start-up both the PC and controller programs therefore go through their tables and check for alphabetic order. If unordered elements are found, the function issues an error message.

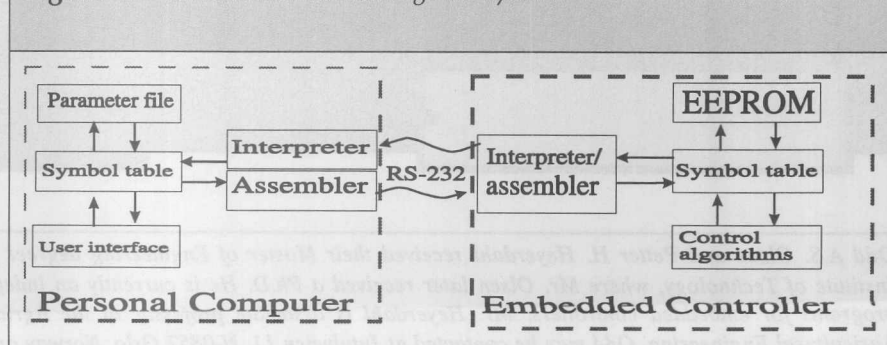
The structure of controller's symbol table is

```
typedef struct {
    char *name;
    int ival;
    char *(*func)(char *);
    int eeOffset;} symTabEntry;
```

name is the name of the parameter, *ival* the value, and the function pointer either a function that can be invoked by a message or a hook for individual processing of received parameters. *eeOffset* is the *EEPROM* address where the parameter is stored. (Parameters that are not saved in *EEPROM* have an offset value of -1.)

The controller's program runs periodically, e.g. every five seconds. The program first reads all measurements and stores the values in the symbol table. The control algorithms then take their values and parameters from the symbol table and calculate responses. The results are stored

Figure 1 Information flow through the system



FIDELITY INVESTMENTS® An Unprecedented Commitment To Applied Technology.

At Fidelity Investments®, we take the best available technologies from the world's leading companies, then integrate them into our open platforms.

Within our Boston development organization, we work to define business initiatives, search for leading-edge solutions, then customize and integrate the technology to take our business to levels beyond our competition's. Whether we are exploring new image processing or voice systems, or defining innovative methods to store and

retrieve data, Fidelity is committed to leading the industry in applied technology.

Fidelity Investments is the largest privately held financial services organization in the country, with assets under management exceeding \$200 billion. We are confident and successful. If you desire to explore the future of applied technology, invest your time in Fidelity. It could be the smartest investment you've ever made.

OPPORTUNITIES CURRENTLY EXIST IN THE SYSTEMS INTEGRATION/FINANCIAL APPLICATION AREA:

Sophisticated Financial Applications Systems Design and Support	Business/Technology Integration and Support
Voice/Data Integration	Voice Processing Architecture
Image Processing Architecture Design and Support	Price Database Design and Support
Stock Market Data Feed Design and Management	

SKILLS REQUIRED: UNIX/SUN, STRATUS, LAN, WAN, C/C++, IMAGE TECHNOLOGY, SYSTEMS INTEGRATION

FOR IMMEDIATE AND CONFIDENTIAL CONSIDERATION...

We're interviewing NOW. So, if you have the skills and ambition to work with the best technologies and developers in the country, mail or fax your resume to:

**JACK KELLY, DEPT. CUJ
C/O FIDELITY INVESTMENTS
82 DEVONSHIRE STREET, MAIL ZONE P2B
BOSTON, MA 02109 FAX: (617) 772-4398.**

In keeping with our unprecedented commitment to applied technology, Fidelity offers outstanding medical and dental care options, work and family resource referral programs, a 100% matched 401(k) plan, as well as pension and profit sharing plans that are company-paid.

**A FULL RELOCATION PACKAGE
IS ALSO AVAILABLE.**



Fidelity Investments is an equal opportunity employer.


```
FindSymbol("pHError")->ival=
FindSymbol("pHsetpoint")->ival
- FindSymbol("pHmeasured")->ival;
```

FindSymbol returns a pointer to the symbol table entry with the given name. However, less typing and a faster program are achieved by initializing pointers to the symbol table entries at start-up:

```
*ppHError= *ppHsetpoint - *ppHmeasured;
```

At start-up the symbol tables are initialized according to the following sequence. First the tables are initialized with the values specified at compile time. The controller then reads values from the *EEPROM* into its table. The PC reads a parameter file, which may change some of the values in its table. The PC program will then transfer those values having flag *TO_EE* set to the controller *EEPROM* and symbol table.

AccuSoft Image Format Library 4.0

Import • Export • Scanning • Conversion • Compression
 Display • Printing • Image-Processing • Special-Effects

Fastest libraries on the market
Guaranteed to read all images in existence

WIN □ DOS □ VBX □ WIN-NT □ OS/2 □ UNIX □ MAC
 WIN-32 Pro Gold □ DOS-32 □ VBX-32 □ and others

When your application demands high quality imaging and top performance, only AccuSoft can deliver. We are the leader in high-performance imaging solutions around the globe. AccuSoft provides the highest quality toolkits with a guarantee that can't be touched!

TIFF

PCX

JPG

TGA

BMP

WMF

- 12 formats supported with auto-detect
- Fastest Group 3, Group 4 and JPEG available!
- Non-standard G3 and G4 files also supported.
- Scanning supported including TWAIN.
- High-speed image display and printing.
- Complete image processing including:
 Rotate, invert, resize, sharpen, blur,
 Roberts Cross, Matrix convolutions,
 Color reduction, edge detection, etc.
- Incredibly easy high-level interface.
- Low-level interface; read & write 1 line at a time.
- All compression algorithms supported.
- Custom versions available.
- 30 day money-back guarantee for 16 bit products.

Call now to order or for more details by fax.

(800) 525-3577
 (508) 898-9662 (FAX)

AccuSoft

High Performance Imaging!

AccuSoft Corp. 112 Turnpike Rd. Westborough, MA 01581 (508) 898-2770

GIF

EPS

WPG

DIB

PICT

DCX

Message Format

The messages between the PC and controller are ASCII strings sent over an RS-232 connection. Before transmission the message assembler adds a checksum. The receiver checks and removes the checksum from the message before passing the message to the message interpreter.

Each message consists of one or more submessages. The submessages and their elements can be separated by spaces and have the following format:

<name><operator>[<value>]

Listing 1 Message interpreter definitions and function prototypes

```
/* parsdef.h */
/* No copyrights claimed */
#define ERROR 1
#define EMPTY 2
#define ASSIGN 3
#define REQUEST 4
#define TO_EEPROM 5
#define IS 6
#define COMMAND 7
#define INTEGER 8
#define NAME 10
#define NAME_ERROR 11
#define OP_ERROR 12
#define VAL_ERROR 13
#define END_ERROR 14
#define POOL_ERROR 15
#define UNDEF_SYMB 16
#define NOT_EEPROM 17
#define UNDEF_FUNC 18
#define MAX_TOKEN_LEN 10
#define MAX_STATEMENTS 20
#define STRINGPOOL 80
#define MESSAGELENGTH 80

typedef struct {
    int command;
    int type;
    char *name;
    int value;
} toDoList;
```

The name field is the symbolic name of a parameter or a command. The operator is a single character as given in Table 1. For messages with an associated value, the value field is the numerical value of the parameter. The name must begin with a letter (a-zA-Z), followed by a number of letters, digits, or underscores (a-zA-Z0-9_). In our implementation the value field is always an integer because all processing in the controller is integer based.

To add other value types the programmer need only define new operators and modify the structures and programs accordingly. A floating-point type is useful in some systems. A string type can also be of use, for example to transfer strings directly to an operator display on the controller. In a multi-processor embedded system, the string type can be used to transfer messages verbatim to subprocessors.

The following message transfers *temp_setp=20* to *EEPROM*, sends *delay=10* to the symbol table, requests the present value of *temp*, and executes the function *pwr_sav*:

```
temp_setp>20 delay=10 temp? pwr_sav!
```

The response might be

```
temp:22 pwr_sav#16
```

The error message indicates that *pwr_sav* was not found in the symbol table of the controller. (Such error messages are hopefully only encountered during program development.) In this case the message indicates a discrepancy between the PC and controller programs.

Message Interpretation

The interpreter parses the received messages into a task list which is handed over to an execution function. The structure of the list records is:

```
typedef struct {
    int command;
    int type;
    char *name;
    int value; } toDoList;
```

command is a value specifying the command, *type* the type of the value (in this case always *INTEGER*), **name* a pointer to the name of the parameter, and *value* the numerical value of the parameter.

Table 1 Message formats and operators

<name> = <value>	Assign value to parameter name
<name> ?	Request the value of parameter name
<name> > <value>	Transfer value to the <i>EEPROM</i>
<name> : <value>	Current value of parameter name
<name> !	Execute name-command
<name> # <value>	Error code value associated with name

In the controller the interpreter and reply assembler are integrated. In the PC, however, parameters are sent through a message assembler and replies decoded in a separate interpreter. The receiver is interrupt driven and executes the interpreter after a full message has arrived. The interpreter places the received values into the symbol table and transfers error messages to the user interface. The PC interpreter resembles the controller interpreter, which I am about to describe.

A Sample Application

This slightly modified version of the controller's interpreter and message assembler will illustrate message handling and symbol table operations. (This version runs on a PC.) Listing 1 presents the definitions and function prototypes; Listing 2 shows the main program and message interpreter.

The main program reads a message from the keyboard. It parses the message, then prints and executes the task list *toDo*. The task list is a list of structures with one entry for each subcommand. The *Parse* function assumes the message is formatted as defined in Table 1. Since the grammar is quite simple, there is no need for recursive descent or other strategies used in compilers. There are several tests that check adherence to the format, and issue error messages if discrepancies are found. (This is most valuable during program development.) Finding an error stops the message interpretation. Inserting an *EMPTY* command terminates the list.

GetNextToken reads the next token in the message string. If the first character encountered is a letter, the token is a name. If it is a


DON'T WASTE YOUR \$\$\$ BUYING SCO'S[®] DEVELOPMENT SYSTEM!

Ready-to-Run Software offers *the* low-cost UNIX[®] software development alternative:


 **ANSI & POSIX compliant GNU C & C++**


 **C, termcap, and curses libraries**

 **Standard header files**

 **GNU gdb debugger**

 **ar, as, ld, nm, ranlib, size, strip . . .**

 **No "per seat" license**

 **30-day money-back guarantee**

 **All in our LanguagePak #1 -- \$275 complete!**

CALL 1-800-743-1723 for a free catalog describing all our products & platforms!

Ready-to-Run Software, Inc.
4 Pleasant St. Forge Village, MA 01886
Phone: (508) 692-9922 FAX/Modem: (508) 692-9990
email: info@rtr.com

In Europe:

User Interface Technologies
P.O. Box 145 Cambridge CB4 1GQ England
Phone: +44.223.302.041 FAX: +44.223.302.042
email: info@uit.co.uk

The Ready-to-Run Software logo is a trademark of Ready-to-Run Software, Inc.



◆ Request 155 on Reader Service Card ◆

Listing 1 *continued*

```

typedef struct {
    char *name;
    int ival;
    char *(*func)(char *);
    int eeOffset; /* -1 if not saved in eeprom */
} symTabEntry;

char *Assign(toDoList *pL, char *pTx);
void DoCommands(toDoList *pL);
symTabEntry *FindSymbol(char *name);
char *GetNextToken(char *begin, char *token, int *type);
void Parse(char *message, toDoList *pL);
void PrintToDo(toDoList *toDo);
char *Request(toDoList *pL, char *pTx);
char *Reset(char *ptr);
char *RunCommand(toDoList *pL, char *pTx);
char *SayError(toDoList *pL, int err, char *cp);
int StoreToken(char **strp, char *token);
int SymCmp(symTabEntry *a, symTabEntry *b);
char *ToEeprom(toDoList *pL, char *pTx);
char *TxBuffer(void);
void WriteEeprom(symTabEntry *pSym, int type);

/* End of File */

```

digit, the token is a value. If the first character is neither a name nor a value, the token is an operator. The function then copies the token into the token string of the parser and sets the *type* parameter according to the type. The size of the token-string (*MAX_TOKEN_LEN*) must be larger than the longest token that can occur. The *type* parameter is set equal to the operator's defined constant.

Parser calls *StoreToken* to save the encountered names in a string pool. This pool is recycled for each message by setting *pPool=stringPool*. The pool size (*STRINGPOOL*) must be large enough to store all the names that can occur in one message. The address of the name is stored in the *name* element of the structure.

The task list *toDo* can now be processed by calling *DoCommands*. This function is essentially a loop which executes the commands through a switch statement. However, it first initializes a pointer to the beginning of the *txBuffer*, where the response message is stored. This buffer might be local to the more hardware-dependent parts of the program and not globally accessible, so this requires a function call to obtain the pointer. All functions called in *Docommands* maintain the buffer pointer by accepting it as an argument and returning the possibly changed value.

SayError writes an error message to the transmit buffer. It receives a pointer to the task list, where it finds the name relating to the error, and the error number.

Listing 2 *The message interpreter and assembler for the embedded controller, implemented on a PC*

```

/* pars.c */
/* no copyrights claimed */
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
#include "parsdef.h"

static toDoList toDo[MAX_STATEMENTS];
static char stringPool[STRINGPOOL];
static char *pPool, *poolEnd;
static char message[MESSAGELENGTH];
static char *messList[] = {
    "EMPTY", "error", "empty", "assign", "request",
    "to_eeprom", "is", "command", "integer", "float",
    "name", "name_error", "op_error", "val_error",
    "end_error", "pool_error"
};

main()
{
    while(1) {
        pPool = stringPool;
        poolEnd = stringPool + STRINGPOOL - 1;
        printf("\nmessage: ");
        gets(message);
        if(message[0] == '.') break;
        Parse(message, toDo);
        PrintToDo(toDo);
        DoCommands(toDo);
    }
}

/* first, the parsing */

void Parse(char *message, toDoList *pToDo)
{
    char *pC, token[MAX_TOKEN_LEN+1];

    int type, i;
    pC = message;
    for(i=0; i<MAX_STATEMENTS; i++) {
        (pToDo+i)->command = EMPTY; /* mark end */
        if(*pC == '\0') {pToDo->command = EMPTY; return;}
        /* get name */
        pC = GetNextToken(pC, token, &type);
        if(StoreToken(&pToDo->name, token) != 0)
            { pToDo->command = POOL_ERROR; return; }
        if((type == ERROR) || (type != NAME))
            { pToDo->command = NAME_ERROR; return; }
        if(*pC == '\0') {pToDo->command = EMPTY; return;}
        /* get operator */
        pC = GetNextToken(pC, token, &type);
        if(type == ERROR)
            {pToDo->command = OP_ERROR; return;}
        pToDo->command = type;
        /* get value */
        if((type == ASSIGN) || (type == TO_EEPROM) ||
           (type == IS)) {
            if(*pC == '\0')
                {pToDo->command = VAL_ERROR; return;}
            pC = GetNextToken(pC, token, &type);
            pToDo->type = type;
            if(type == INTEGER)
                pToDo->value = atoi(token);
            else {pToDo->command = VAL_ERROR; return;}
        }
        else pToDo->type = INTEGER;
        if(pToDo->command == EMPTY) return;
        pToDo++;
    }
    pToDo->command = END_ERROR;
    return;
}

char *GetNextToken(char *begin, char *token, int *type)
{

```

Wind River Systems Needed a Real-Time Development Environment for Windows. They Selected MULTI™ for VxWorks®.

*In a multi-tasking environment,
each task can have its own window.*

The screenshot displays the MULTI development environment with several windows open:

- Task Window:** A table listing tasks with columns: Tid, Name, Entry, Status, Priority.

Tid	Name	Entry	Status	Priority
3da354	tLogTask	logTask	PENDING	0
3d5fc8	tNetTask	netTask	PENDING	50
3bab2c	tRlogind	rlogind	PENDING	2
3b8c88	tTelnetd	telnetd	PENDING	2
3b7770	tPortmapd	portmapd	PENDING	100
3b6258	tRdbTask	rdbTask	DELAY	20
3b3da0	tFtpdTask	ftpdTask	PENDING	55
3ac1b0	tShell	shell	PENDING	1
3a1448	tRdbRun		PENDING	100
39c498	tRdbRun	windDemo SUSPEND	100	
3974e8	tRdbRun	windDemo SUSPEND	100	
392538	tRdbRun	windDemo SUSPEND	100	
- Source Code Window (windDemo.c):** Shows C code for a task named 'windDemo'. It includes comments and code for initialization, loop, and object creation.


```

void windDemo
(
    int iteration
)
{
    int loopCount = 0;

    #ifdef STATUS_INFO
    printf ("Entering windDemo\n");
    #endif /* STATUS_INFO */

    if (iteration == 0)
    {
        0x3a3d76: TST.L D2
        0x3a3d78: BNE windDemo#
        iteration = 10000;
        0x3a3d7a: MOVE.L #10000,D2

        /* create objects used by the child tasks */
        msgGld = msgGCreate (MAX_MSG, MSG_SIZE);
        semId = semBCreate (SEM_Q_PRIORITY, SEM_Q_SIZE);
        wldId = wdCreate ();

        windDemo = taskSelf ();

        FOREVER
        {
            /* set default iteration = 10000;

            /* create objects used by the child tasks */

```
- Debugger Windows:** Multiple windows showing assembly code, registers, and memory addresses, with a 'STOPPED INSIDE' status indicating a breakpoint.

You Can Use MULTI for Your Real-Time Applications Too.

MULTI: The Most Comprehensive and Completely Integrated Embedded Solution

MULTI is a multi-language embedded development environment featuring source-level debugging, execution profiling, memory leak detection, graphical class browser, program builder, built-in editor, and source code control.

MULTI is tightly integrated with other code development tools such as compilers, assemblers, linkers, librarians, and target execution environments including simulators, ROM monitors, and in-circuit emulators.

MULTI incorporates the Green Hills optimizing compilers to achieve a development environment with a common user interface for 5 languages and all popular 32 and 64-bit target microprocessors across a wide range of host development platforms. Green Hills compilers use 100+ optimizations to produce unmatched code for your toughest applications.

Languages

• C • C++ • Fortran • Pascal • Ada

Host Development Platforms

• Microsoft Windows • Unix Workstations • VAX/VMS

Target Microprocessors

• 680x0/683xx • 80386/80486 • SPARC
• R3000/R4000 • i960 • V810

**Call (800) 500-2580 Today
for a Free MULTI Demo!**

Oasys

One Cranberry Hill
Lexington, MA 02173
Tel: (617) 862-2002
Fax: (617) 862-3073

Green Hills Software, Inc.

510 Castillo Street
Santa Barbara, CA 93101
Tel: (805) 965-6044
Fax: (805) 965-6343



GSS*CGI Graphic Tools now from EMATEK

Soon available on all platforms:

MS-Windows, Windows NT, OS/2 PM, Solaris, UNIX on PCs and workstations.

Your application written with the GSS*CGI Graphic Tools
will be portable to all platforms.

*We comply
with standards!*

GSS*GDT

CGI-Standard

Graphics Development Toolkit enables you to develop applications in a device and system independent way through the help of device-specific drivers based on the ISO CGI Standard. GSS*GDT consists of a library with more than 160 callable C and FORTRAN functions and is compatible with the graphics development tools from IBM and SCO.

GSS*GDT is available for DOS, MS-Windows, Windows NT, OS/2 PM, Interactive Unix, Onsite Unix SVR 4.2 and Solaris.

Versions for SCO Unix, UnixWare and HP Workstations will be soon available.

GSS*GKS

GKS-Standard

Graphical Kernel System is a C and FORTRAN function library that enables you to develop portable graphic applications which include e.g. user interaction, coordinate transformation and object segmentation, based on the ISO GKS Standard. GSS*GKS, which is installed in large quantities on the DOS platform and has been proved successful for years, is now available for the graphical user interfaces and therefore offers the software developer a smooth transition to the new windowing systems.

GSS*GKS is available for DOS, MS-Windows, Windows NT, OS/2 PM, Interactive Unix, Onsite Unix SVR 4.2 and Solaris.

Versions for SCO Unix, UnixWare and HP Workstations will be soon available.

GSS*EVT

EMATEK Vectorfont Toolkit enables you to integrate vectorfonts into your application. Supported font formats are PCL5, PostScript Type 1, TrueType and Bitstream Speedo. GSS*EVT offers a variety of additional elements and attribute functions like character height and gap, rotation and italicize angle, fill area pattern and outline-width, shadow, background boxes, leader and underlining as well as sub- and superscripting for scientific purposes. GSS*EVT is an add-on for GSS*GDT, GSS*GKS, MS-Windows SDK and will be ported to all popular graphical user interfaces.

GSS*GCT

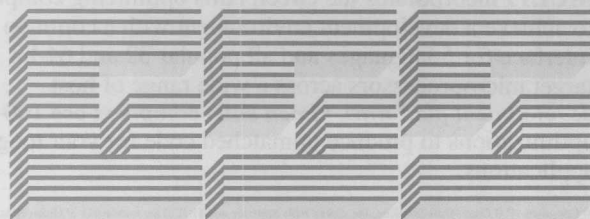
Graphics Charts Toolkit provides you with high-level functions to integrate presentation graphics into your application. With only a few calls you can output pie, bar, line, step, schedule or text diagrams on a display or printer. Each part of the chart can be altered through the related attribute functions. GSS*GCT is an add-on for GSS*GDT and GSS*GKS and will be ported to all popular graphical user interfaces, thus providing portable presentation graphic capabilities for all supported systems.

*Comply too with standards
to ensure the long-term value of your applications!*

◆ Request 219 on Reader Service Card ◆

EMATEK

Subbelrather Strasse 17 · D-50823 Cologne, Germany
Phone: +49-221-512074 · Fax: +49-221-529666



A sign of Quality for over 12 Years

The functions *Assign*, *Request*, *ToEeprom*, and *RunCommand* each begin by getting a name from the task list and looking it up in the symbol table. If the name isn't found, they write an error message to the transmit buffer.

Listing 2 continued

```
char *pC, *pT;
int i;
pC= begin;
pT= token;
*token= '\0';
*type= EMPTY;
if(*pC == '\0') {return pC;}
/* remove leading spaces */
while(*pC==' ') pC++;
if(*pC=='\0') return pC;
if(isalpha(*pC)) { /* name */
    *type= NAME;
    *pT+= *pC++;
    for(i=1; i<MAX_TOKEN_LEN; i++) {
        if((isalpha(*pC)) || (isdigit(*pC))
            || (*pC=='_')) *pT+= *pC++;
        else {
            *pT= '\0';
            return pC;
        }
    }
    *type= ERROR;
    return pC;
}
else if((isdigit(*pC)) || (*pC=='+' ||
    (*pC=='-')) { /* number */
    *pT+= *pC++;
    for(i=1; i<MAX_TOKEN_LEN; i++) {
        if(isdigit(*pC)) *pT+= *pC++;
        else {
            *pT= '\0';
            *type= INTEGER;
            return pC;
        }
    }
    *type= ERROR;
    return pC;
}
else if(*pC== '=') *type= ASSIGN;
else if(*pC== '?') *type= REQUEST;
else if(*pC== '>') *type= TO_EEPROM;
else if(*pC== ':') *type= IS;
else if(*pC== '!') *type= COMMAND;
else {*type= ERROR; return ++pC;}
return ++pC;
}

int StoreToken(char **ppName, char *token)
{
    int length;
    *ppName= pPool;
    length= strlen(token)+1;
    if((pPool+length)>=poolEnd) return -1;
    strcpy(pPool, token);
    pPool+= length;
    return 0;
}

/* and now, the action */

static symTabEntry symTab[]= {
```

Learn C++ & Windows™-Based Programming-Simply, Quickly

The OML Learning Series is the most cost effective way to learn C++ and Windows-based programming in the privacy of your home or office. The highly interactive graphical interface provides an intuitive and speedy OOP learning environment on your PC*. Numerous exercises and examples help reinforce key concepts. OML offers:

- ☐ **C/C++ Series** (5 courses)
- ☐ **OOA and OOD Series** (2 courses)
- ☐ **Visual Series** (MFC 2.0.)
- ☐ **Other Series** (call)
- ☐ **Training/Consulting** (call)

We believe the best way to convince you that our courses are the perfect tool for your object-oriented training is to use it personally. We have created a special evaluation package that includes our complete curriculum. With this package, **you may review any part of any of our courses**, up to 25% of each course. The evaluation package is priced at \$20.00**.

For a limited time, each course is available at the special price of **\$59.95** (regular price \$129.95). Ask for our volume discount and student price. To order our courses or our evaluation package, call today:

1(800) 6789-OML

VISA/MC/AMEX accepted, s/h extra.

* Requires MS-Windows or OS/2 2.0 VGA, 386 or 486, and a mouse.

** This \$20.00 fee will be applied toward the purchase of your first OML course.

Windows is a registered trademark of Microsoft Corporation

4165 Thousand Oaks Blvd.
Suite 225
Westlake Village, CA 91362
Phone: (805) 373-8111
Fax: (805) 373-8116



◆ Request 106 on Reader Service Card ◆

The *Assign* function inserts the value in the symbol table. The *Request* function reads the value from the symbol table and generates a response submessage. *ToEeprom* puts the value into the symbol table and checks if the parameter has a place in the EEPROM. If *eeOffset* is greater than or equal to zero, it writes the value. If not, it issues an error message. *RunCommand* executes the program pointed to in the symbol structure. If the pointer is NULL, *RunCommand* generates an error message.

Assign, *Request* and *ToEeprom* all check for a function hook before returning. If they find one they call it. *Assign*, *Request*, and *ToEeprom* also maintain the transmitter buffer pointer so they can write to the buffer.

The program was compiled with Borland TurboC++ version 3.0. The library functions are all UNIX and ANSI defined, so using other compilers should not present any problem.

Final Thoughts

This symbolic data transfer scheme enabled us to implement a complete control system from scratch in about one programmer-month. When we started we had no clear picture of what the final result would be, so parameters were constantly added and removed as we tried different control strategies. By using the described method, we were able to respond quickly to these changes. This arrangement also made it easier to add auxiliary parameters for debugging. There are of course faster ways to transfer data, but the kinds of processes maintained by embedded controllers are often slow enough that a minor increase in response time isn't significant. □

Real-time Multi-tasking Full C source Code Included

The Tics Realtime multi-tasking kernel is a linkable C library that allows C functions to run as preemptive concurrent tasks. Developed for professional software developers who need a capable full-featured real-time multi-tasking development system. Tics can run with MS-DOS or stand-alone on an embedded target. No royalties. Tics also includes a multi-tasking COM port library for multi-tasking serial communications.

- C source code is included - fully documented and complete.
- Includes our new book "The Art of Real-time Programming".
- Tics is written entirely in C and is portable to virtually any microprocessor.
- Includes integrated COM port library for multi-tasking RS-232 applications.
- The PC timer chip is reprogrammed to user specified granularity.
- Preemptive, cooperative, or time-sliced scheduling, configurable on a task by task basis.
- No restriction on the number of tasks.
- Stack size is variable on a task by task basis.
- Priorities can be changed dynamically.
- Tasks may be created dynamically.
- The development system also includes TinyTics, a small round robin kernel that is ideal for micro-controllers.
- Tics requires less than 8K bytes of code space and is ROMable.
- Three types of high speed timers: in-line pause, one-shot timers, and time critical periodic timers.

- Differential timer management system allows for an unrestricted number of concurrent timers without increasing time spent in the timer isr.
- Inter-task communication using message queues or high speed mailboxes.
- Critical region management.
- High speed memory management system. Unrestricted number of memory pools.
- Priorities on tasks and messages provide great flexibility. Priorities on messages means that high priority messages go straight to the front of the queue.
- Optionally, cooperative tasks may share a common stack so that hundreds of tasks can run with the overhead of a single stack.
- Object oriented. Create task instances with their own separate instance data. Can run with C++ for real-time OOP.
- Messages or mail can be sent from within an isr.
- Time out option while waiting for a message.
- Includes manual and source code for sample applications that include dials, gauges, data acquisition, display, and more.



790 Lucerne Drive Suite 78
Sunnyvale, CA 94086
(408) 723-2200

Price: \$499

Shipping & Handling: \$5, \$15 Int.

◆ Request 308 on Reader Service Card ◆

Listing 2 continued

```

{"reset" , 0 , Reset, -1},
{"status" , 35 , NULL , -1},
{"temp" , 16 , NULL , 20}
};

void PrintToDo(toDoList *toDo)
{
    while(toDo->command != EMPTY) {
        printf("\n %s %s %s",
            messList[toDo->command],
            messList[toDo->type], toDo->name);
        if(toDo->type==INTEGER)
            printf(" %d", toDo->value);
        toDo++;
    }
}

int SymCmp(symTabEntry *a, symTabEntry *b)
{
    return strcmp(a->name, b->name);
}

symTabEntry *FindSymbol(char *pName)
{
    symTabEntry dummy, *p;
    dummy.name= pName;
    p= (symTabEntry*) bsearch(&dummy, symTab,
        sizeof(symTab)/sizeof(symTabEntry),
        sizeof(symTabEntry),
        (int (*)(const void*, const void*))SymCmp);
    return p;
}

char *Reset(char *pC)
{
    printf("\nExecuting 'Reset'");
    return pC;
}

void DoCommands(toDoList *pToDo)
{
    char *txBuffer, *pTx;
    txBuffer= pTx= TxBuffer();
    *pTx= '\0';
    while(pToDo->command != EMPTY) {
        switch(pToDo->command) {
            case ERROR:
            case NAME_ERROR:
            case OP_ERROR:
            case VAL_ERROR:
            case END_ERROR:
            case POOL_ERROR:
                pTx= SayError(pToDo, pToDo->command, pTx);
                break;
            case ASSIGN:
                pTx= Assign(pToDo, pTx);
                break;
            case REQUEST:
                pTx= Request(pToDo, pTx);
                break;
            case TO_EEPROM:
                pTx= ToEeprom(pToDo, pTx);
                break;
            case COMMAND:
                pTx= RunCommand(pToDo, pTx);
                break;
            default:
                break;
        }
        ++pToDo;
    }
    printf("\ntxBuffer: '%s'\n", txBuffer);
}

```

```

char *SayError(toDoList *pToDo, int err, char *pC)
{
    char str[MAX_TOKEN_LEN+1], *pS;
    pS= pToDo->name;
    while(*pC++=*pS++);
    pC--;
    *pC+= '#';
    sprintf(str,"%-d", err);
    pS= str;
    while(*pC++= *pS++);
    return --pC;
}

char *Assign(toDoList *pToDo, char *pTx)
{
    symTabEntry *pSym;
    pSym= FindSymbol(pToDo->name);
    if(pSym==NULL) {
        pTx= SayError(pToDo, UNDEF_SYMB, pTx);
        return pTx;
    }
    if(pToDo->type==INTEGER) pSym->ival= pToDo->value;
    if(pSym->func!=NULL) pTx= (*pSym->func)(pTx);
    return pTx;
}

char *Request(toDoList *pToDo, char *pTx)
{
    symTabEntry *pSym;
    char *pS, str[MAX_TOKEN_LEN+1];
    pSym= FindSymbol(pToDo->name);
    if(pSym==NULL) {
        pTx= SayError(pToDo, UNDEF_SYMB, pTx);
        return pTx;
    }
    pS= pToDo->name;
    while(*pTx++=*pS++);
    pTx--;
    *pTx+= '=';
    if(pToDo->type==INTEGER)
        sprintf(str,"%-d", pSym->ival);
    pS= str;
    while(*pTx++=*pS++);
    if(pSym->func!=NULL) pTx= (*pSym->func)(pTx);
    return --pTx;
}

char *ToEeprom(toDoList *pToDo, char *pTx)
{
    symTabEntry *pSym;
    pSym= FindSymbol(pToDo->name);
    if(pSym==NULL) {
        pTx= SayError(pToDo, UNDEF_SYMB, pTx);
        return pTx;
    }
    if(pSym->eeOffset<0) {
        pTx= SayError(pToDo, NOT_EEPROM, pTx);
        return pTx;
    }
    if(pToDo->type==INTEGER) {
        pSym->ival= pToDo->value;
        WriteEeprom(pSym, INTEGER);
    }
    if(pSym->func!=NULL) pTx= (*pSym->func)(pTx);
    return pTx;
}

char *RunCommand(toDoList *pToDo, char *pTx)
{
    symTabEntry *pSym;

```

```

    pSym= FindSymbol(pToDo->name);
    if(pSym==NULL) {
        pTx= SayError(pToDo, UNDEF_SYMB, pTx);
        return pTx;
    }
    if(pSym->func!=NULL) pTx= (*pSym->func)(pTx);
    else {
        pTx= SayError(pToDo, UNDEF_FUNC, pTx);
        return pTx;
    }
    return pTx;
}

char *TxBuffer(void)
{
    static char buffer[200];
    return buffer;
}

void WriteEeprom(symTabEntry *pSym, int type)
{
    if(type==INTEGER) printf(
        "\nTo EE, type: %d off: %d val: %d\n",
        type, pSym->eeOffset, pSym->ival);
    return;
}

/* End of File */

```

TLIBTM Version Control **For DOS, OS/2 and Windows-NT**

• The experts loved TLIB 4:

"...amazingly fast... TLIB is a great system." **PC Tech Journal**
"TLIB has features and power to spare... TLIB is easy to use and the fastest of the reviewed packages." **Computer Language**
"I will not program without it." **Uptime Magazine**

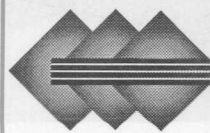
• Now TLIB 5.0 adds:

Automatic branching. Automatic version labeling across branches. User defined **promote** structures, for staged development. Exclusive whole-level **change migration** for customized software. N-way-tree version numbers. Branch and full locking. OS/2 & NT support.

• Plus the features they loved in TLIB 4:

Check-in/out locking. Branching, for parallel development. Keywords. Full binary file support (does not depend upon CRs in the file like other products). Wildcard and list-of-file support; can create lists by scanning source code for includes. Can merge (reconcile) multiple simultaneous changes and undo intermediate revisions. Network and WORM optical disk support. Mainframe-compatible delta generator for Pansophic, ADR, IBM, Sperry formats. Includes integrated PD MAKE by L. Dyer; also integrates with OpusTM MAKE, SlickTM MAKE, others.

MS-DOS \$139, OS/2 & NT (with MS-DOS) \$195+ shipping.
 5 station network: MS-DOS \$419, OS/2 \$595. Call for other sizes.



Burton Systems Software

PO Box 4156, Cary, NC 27519 **(919) 233-8128**
 FAX: 233-0716

◆ Request 430 on Reader Service Card ◆

POWERBUILDER DEVELOPERS WANTED

We're looking for senior application developers and technical architects to join our rapidly expanding client/server team. To succeed, you will have at least 6 months of relevant experience which must include at least 6 months of PowerBuilder development within an SQL compliant relational database.

Wellington Associates
Fax: 800-555-8333

CLIENT/SERVER EXPERIENCE

Unique opportunities are available for programmers experienced in implementing Client/Server systems. Applicants must have a four-year technical degree, four to six years' development experience, and proficiency with both Oracle and PowerBuilder. Benefits include tuition reimbursement and relocation assistance.

PROGRAMMERS/ANALYSTS

Information consulting firm has opportunities throughout the East as well as for professionals with strong technical backgrounds in systems analysis and methodology, as well as experience with PowerBuilder. Benefits include excellent salary, health insurance, and a 401(k) plan.

ANALYSTS/PROGRAMMERS

Builder P/A's
DBA's 50-60k
40-60k
45-65k
40-50k
40-50k
40-50k
40-50k
35-45k
35-45k

Wanted: Desktop
Developers Interested
In Getting
To Client/Server.

Credentials for sale: \$249.



When it
client/server
of Power
by glance

almost any trade journal.

both called it "Product

two Readers' Choice

Sources honored it with

But PowerBuilder

when you peruse the

For there you can

programmers with experience

proven client/server development tool.

So you'll be glad to know that PowerBuilder, previously available to developers with corporate IS needs and resources, is now available to desktop database developers as well.

Thanks to new PowerBuilder Desktop. A version of PowerBuilder created specifically for your standalone and networked desktop-class databases.

Similar to PowerBuilder Enterprise, this edition lets you create applications using a common object technology.

... lets your programs, as well as your
... any

For only \$399, attend our one-day *Getting to Client/Server* training class and take home your own copy of PowerBuilder™ Desktop. Look for the city nearest you and call the toll-free number listed below for dates and details.

Atlanta
Boston
Calgary
Chicago
Cincinnati
Dallas
Denver
Detroit
Houston

Indianapolis
Los Angeles
Minneapolis
Nashville
New Orleans
New York
Orlando
Philadelphia
Pittsburgh

Portland, OR
Salt Lake City
San Francisco
San Jose
Seattle
St. Louis
Toronto
Vancouver
Washington, D.C.

1-800-946-3500

Powersoft™



digital™

Training classes co-sponsored by Digital Equipment Corporation and Powersoft.

reseller or stop by CompUSA
and we'll tell you the nearest location and date of our one-day comprehensive client/server training class that includes your own copy of PowerBuilder Desktop for just \$399.

After all, the price of becoming a client/server developer is a lot lower than the price of not becoming one.

Powersoft™

Building on the power of people.

Powersoft Corporation, 70 Blanchard Road, Burlington, MA 01803
Powersoft Europe, Thames House, 1 Bell Street, Maidenhead, Berkshire, SL6 1BU, United Kingdom

All trademarks and registered trademarks are property of their respective owners. Prices listed do not include sales tax, shipping and handling. 30-day money back guarantee.

**“Don't
Just
Tell
Me...
Show Me.”**

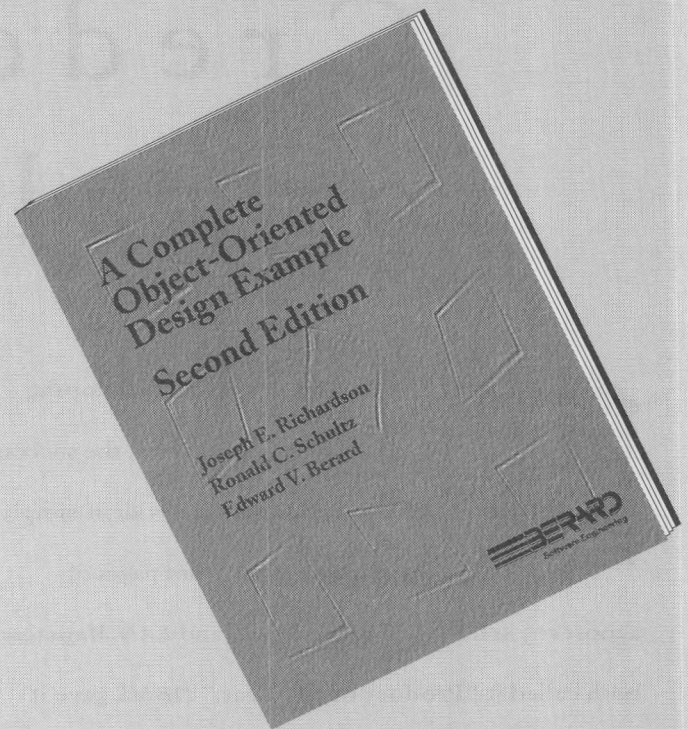
They say everybody loves an example. Since 1982, Berard Software personnel have trained thousands of people in object-oriented software engineering. Each client has had his or her own specific concerns, needs, and wants. But one thing they have all had in common is the desire to see a fully-worked object-oriented design example.

Our software engineers sought out an example problem that was straightforward but realistic. While the problem itself was important, the approach to solving it had to take center stage. The result? a *complete* object-oriented design example:

- The solution is complete — from statement of the problem to design documentation to delivered source code — nothing is left as “an exercise for the reader;”
- A significant amount of the thinking behind various design decisions, including discussions of design alternatives, is presented;
- The example shows solutions in two different programming languages, i.e., C++ and Smalltalk;
- The approach taken is generally programming-language independent, and easily adaptable to the object-oriented programming language of your choice.

This book, including all documentation and source code, contains over 250 clearly organized pages with informative graphics to aid the reader throughout. But we at Berard Software wanted to make your

“OK.”



package even more complete. Available in either Macintosh or DOS formats and in both Smalltalk and C++ is the machine-readable source code used in the example... Now what could be more complete than that?

A Complete Object-Oriented Design Example is **Available Now.** Use the form below or contact Russ Hopler at (301) 417-9884.

<input type="checkbox"/>	A Complete Object-Oriented Design Example (ISBN 1-881974-01-4): U.S. \$50.00 per copy
<input type="checkbox"/>	Machine-readable source code used in <i>OOD Example</i> (U.S. \$15.00; Circle one: DOS MAC)
<input type="checkbox"/>	Sales Tax (5%, Maryland Residents Only — U.S. \$2.50 per book, \$0.75 per diskette)
<input type="checkbox"/>	Shipping and Handling: Add US: \$5 per book, \$1 per diskette (if shipped separately). Canada: US \$10; Europe: US \$20; Australia/Far East: U.S. \$25
<input type="checkbox"/>	VISA <input type="checkbox"/> AMERICAN EXPRESS \$ _____
<input type="checkbox"/>	DISCOVER <input type="checkbox"/> MASTERCARD <i>Total Enclosed</i>

Name & Title: _____
 Address: _____
 City, State, Zip: _____
 Credit Card #: _____ Exp. Date _____
 Signature _____ Date _____
 Telephone: Area Code (_____) _____

BERARD Software Engineering

902 Wind River Lane, Suite 203, Gaithersburg, Maryland 20877
 Phone: (301) 417-9884 • Fax: (301) 417-0021 • Internet: info@bse.com

ROMLDR, an Embedded System Program Locator

Charles B. Allison

MS-DOS software development tools such as C compilers and debuggers have become marvelously sophisticated and useful. They can also provide a cost-effective means for developing embedded systems. The purpose of this article is to introduce one of the aspects of using a high performance DOS-based C compiler for embedded systems, that of relocating code and data segments. I provide a program, *ROMLDR*, which can modify a program from the MS-DOS EXE format to a located binary file format.

ROMLDR's principle purpose is to adapt DOS-based C compiler output for use in EPROM-based embedded systems. Other uses include BIOS extensions, EPROM-based MS-DOS applications, and relocation of MS-DOS programs in PC memory, such as above the 640k MS-DOS memory limit. *ROMLDR* can also be used with EXE files generated by languages other than C.

ROMLDR was written using Borland C 3.1. It should be possible to use either Borland Turbo C or Microsoft C to make the *ROMLDR* program and to use for embedded programs. I have tested *ROMLDR* only with the Borland C 3.1 compiler.

Embedding DOS-Compiled Programs

To use DOS-compiled programs in EPROM-based embedded systems, you must provide several additional components, as well as resolve some unique programming issues. A number of the required components, such as the startup code, depend on the com-

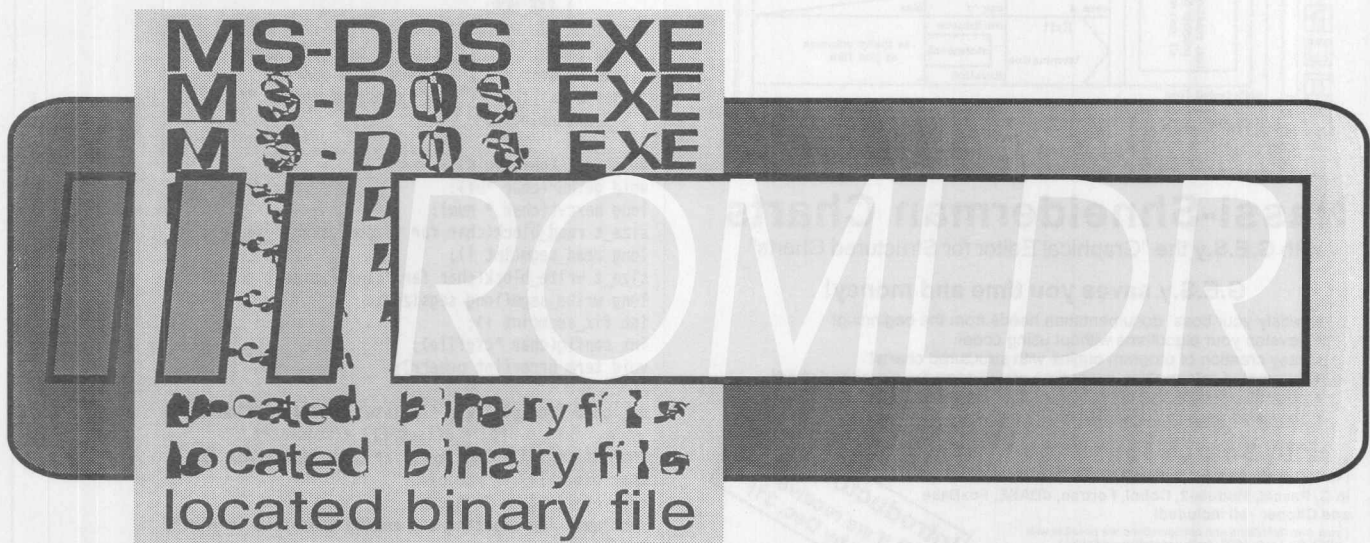
piler, target hardware, and application. (See the sidebar "Coding for Embedded Applications" for a brief discussion on embedded code requirements.)

Once you have attended to all these details, you can program, compile, and link the various program files into an MS-DOS EXE file. However, you can't just burn your EXE file into EPROM and go. You must explicitly perform a step that MS-DOS performs implicitly when it loads an EXE file.

MS-DOS modifies programs with the EXE extent when it loads them into memory. This modification, often referred to as a fix up (also known as the *locate* function), consists of modifying the program's segment values for the actual memory address where it is to run. By performing fix ups, DOS can load an executable almost anywhere in real-mode memory. (DOS can load small programs with the COM extent anywhere and run them without modification.)

DOS performs fix ups by adding the file's load-address segment value to all segment addresses stored in the code and data areas of the program.

Unlike DOS programs, most embedded systems programs reside in and execute from EPROM. Embedded system locators must perform fix ups prior to placing the program in EPROM and must take into account that variables will be located in RAM, through their initialized values for startup are still in EPROM.



Charles Allison has been working with microprocessor hardware and firmware in embedded systems since 1976. He has a Bachelor of Science degree in physics and a Master of Business Administration degree. Charles has a microprocessor consulting business, Allison Technical Services, where he has been developing embedded control and monitoring products for clients since 1984. Charles can be reached through CompuServe 71005,1502, his BBS/FAX line at (713)-777-4746 or his company, ATS, 8343 Carvel, Houston, TX 77036.

Locating for Embedded Systems

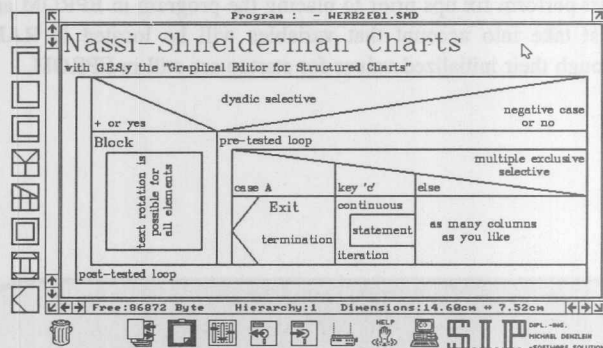
Before I present my program in detail, I want to outline the major parts of the location process. I will first describe the structure of a DOS EXE file, and how that structure is reflected in my code. Next, I will describe the location process.

EXE File Format

The EXE file consists of several sections, including a header, program code, data initialization values, and optional program debug information. Refer to structure *EXE_HDR* at the beginning of Listing 1 for the layout and definitions of the various parameters. The header section consists of several parameters which define the size of the file and the size of the header. Following these parameters is a section of fix-up *far* pointers.

Each of these pointers targets a location in the program code containing a segment address value that must be modified with the correct load address. These pointers are stored in standard 80x86 *segment:offset* format relative to the beginning of the code section. The pointers' segment values are derived from the MAP file segment table by using the top four hexadecimal digits from the beginning address listed for each segment. (MAP files are generated by the compiler. MAP file segments consist of the top four hexadecimal digits of the beginning address.)

There are *num_reloc* fix-up pointers in the header section. These pointers begin at the offset *off_reloc* from the beginning of the header. (Note that fix-up pointers may not be sorted by address as they occur in the EXE file.) Following the header is the program's



Nassi-Shneiderman Charts

with G.E.S.y the "Graphical Editor for Structured Charts"

G.E.S.y saves you time and money!

- + satisfy your boss' documentation needs from the beginning!
- + develop your algorithms without using code!
- + easy creation of program outline with structured charts!
- + automated reformatting every time you change the structured chart!
- + powerful tools like a clipboard, preview, text editor, chart administration!
- + seamless pass to the implementation phase...

...with Source Text Skeleton Output

(all control elements and key words with text in comments)
in C, Pascal, Modula-2, Cobol, Fortran, dBASE, FoxBase
and Clipper - all included!
(your own definitions and configurations are possible with
ASCII file. Up to 100% code generation available!)

SIP
Griesackerstr. 15
D-96117 Memmelsdorf
GERMANY
CompuServe 100120,2601

Tel.: +49-951-43489
FAX: +49-951-420514



G.E.S.y V2.1 \$198

Fax or write for FREE demo!

Please note disk format!

International: add \$10 for shipping and handling to all prices.

◆ Request 230 on Reader Service Card ◆

Listing 1 The EXE locator program

```
/* ROMLDR.C EXE locator program
   written by: Charles B. Allison
   last change: 11-3-93 */

#include <sys/stat.h>
#include <iio.h>
#include <dos.h>
#include <stdio.h>
#include <stdlib.h>
#include <alloc.h>
#include <conio.h>
#include <ctype.h>
#include <fcntl.h>
#include <string.h>
#define BC 1
/* BC is Borland c, else assume Microsoft */
typedef struct {
    unsigned sig; /* signature = 4d5ah */
    unsigned lst_sec_lng;
        /*length of last sector in file modulo 512*/
    unsigned file_size;
        /* size of file in 512 byte pages includes hdr*/
    unsigned num_reloc;
        /* number of relocation items */
    unsigned hdr_siz;
        /* # of 16 byte paragraphs in header */
    unsigned min_ld_para;
        /* min # of paragraphs above load file */
    unsigned max_ld_para;
        /* max # of paragraphs requested by file */
    unsigned disp_stack_seg;
        /* rel displacement of stack segment */
    unsigned sp;
        /* contents of stack ptr on entry to prog */
    unsigned chksum; /* check sum for file */
    unsigned ip; /* beginning instruction ptr */
    unsigned rel_cs_seg; /* relative cs segment */
    unsigned off_reloc;
        /* offset to 1st relocation item typ. 1e */
    unsigned over_lay; /* overlay number */
    unsigned rsrvd; /* ?? reserved ?? */
} EXE_HDR;
/* relocation item format is seg:off location
   relative to the beginning of the code section */
struct MP_TBL {
    long addr; /* segment beg. address */
    long haddr; /* segment high addr*/
    char class[12]; /* class of object */
} mactable[120];

void sort_table(void);
void gethdr(char *bf);
long hexcvt(char * num);
size_t read_block(char far *segbuf, size_t segsz);
long read_seg(int i);
size_t write_block(char far *segbuf, size_t segsz);
long write_seg(long segsize);
int fix_seg(int i);
int config(char *cfgfile);
void term_error(int numerr);

unsigned (*ch_ptr)[2]; /* pointer to translation table
                        [0] = offset, [1] = segment*/
char mfile[14] = "rom.map"; /* dummy file names */
char bfile[14] = "rom.bin";
char efile[14] = "rom.exe";
/*columns for starting and ending addresses in MAP */
#define LCOL 1
#define HCOL 8
#ifdef BC
#define MAPCOL 41;
```

```

/*bc map file class column*/
#else
#define MAPCOL 45;
/*ms map file class column */
#endif
int class_loc = MAPCOL;
FILE *mapfile,*exefile,*binfile;
#define BUF_SIZE 60
char mapstring[BUF_SIZE];
long fsize; /*number of bytes in exe file */
int nsegs; /* number of segments in map */
unsigned romsadr = 0xf000,ramsadr=0x40;
char header[10000];
EXE_HDR *filhdr = (EXE_HDR *)header;
char far *seg_buffer;
int next_fix = 0;
char ram_class[15] = "FAR_DATA";
unsigned ramdata; /* beginning ram segment*/
/* ***** main ***** */
int main(int argc,char *argv[])
{
    int r_class_flag=1,i;
    long tmp,ssize;

    if((seg_buffer = (char far *) farmalloc(0x10000L))
        == NULL) term_error(0);
    if(argc == 1) term_error(-1); /*any cfg file name? */
    config(argv[1]);
    i=0;
    while (fgets(mapstring,BUF_SIZE,mapfile))
    {
        /* process map file from mapstring input */
        /* mactable[1] - n contains the segments -
        class STACK should be last one */
        /* ends with i having n+1 segments */
        if( (int)strlen(mapstring) > class_loc+1)
        {
            /* get rid of \n at end of string */
            mapstring[(int)strlen(mapstring)-1]='\0';
            if((tmp = hexcvt(&mapstring[LCOL])) >= 0)
            {
                mactable[i].addrs = tmp;
                mactable[i].haddrshexcvt(&mapstring[HCOL]);
                &mapstring[class_loc]);
                strcpy(mactable[i].class,
                    &mapstring[class_loc]);
                if(r_class_flag)
                if(strcmp(&mapstring[class_loc],ram_class)==0)
                { /*set it to first class occurrence*/
                    ramdata = (mactable[i].addrs) >>4;
                    r_class_flag = 0;
                }
                printf("\n Segment %4.4lx Class %s",
                    mactable[i].addrs/16,mactable[i].class);
                i++;
            } /* end - if hexcvt */
        } /* end if strlen */
        if(i>=119) break; /* error too many segments */
    } /* end while */
    if(feof(mapfile))
        printf("\nend of file\n");
    else
        printf("\nerror reading map file\n");
    nsegs = i-1; /* number of segments [1 to nsegs] */
    gethdr(header); /* read in the exe header info */
    /* size of object section of file */
    fsize = (long) ((512L * (filhdr->file_size-1)) +
        filhdr->lst_sec_lng - 16L * filhdr->hdr_siz);
    printf("\nStartup Address %4.4x:%4.4x\n",romsadr+
        filhdr->rel_cs_seg,filhdr->ip);
    printf("Rom Size %lx\n",fsize);
    /* process the exe file header - sort fix ups */
    sort_table();
    /* read in exe file by segment
    do fixups from map table and

```

```

write it to output file */
for(i=0;i < nsegs;i++)
{
    if((ssize = read_seg(i)) > 0L )
    { /* ignore 0 length segments */
        fix_seg(i);
        write_seg(ssize);
    }
}
/* done - end the program */
farfree(seg_buffer);
fcloseall();
return 0;
}
/* ***** */
/* qsort routine for far pointers */
int cmp_ptr(const void *a, const void *b)
{
    long vala,valb;
    vala=((long)((unsigned *)a)[0])+
        (((unsigned *)a)[1]<<4);
    valb=((long)((unsigned *)b)[0])+
        (((unsigned *)b)[1]<<4);

    vala -= valb;
    if(vala < 0) return -1;
    if(vala > 0) return 1;
    return 0;
}

```

Conversion Solutions for C and C++ Programmers

■ FOR_C®

Converts standard FORTRAN and many VAX, PRIME and IBM-VS extensions into ANSI C! Features C-style preprocessing for easy mapping of FORTRAN library calls, C prototyping for enhanced function call translations, extensive static analysis and error checking plus compiler-like performance. Translated code is extremely *readable* and *maintainable*!

■ FOR_C++®

The only FORTRAN to C++ translator available! Translations support C++ string and complex data classes for good code readability. Generates C++ function prototypes, checks function calls for consistent usage, optimizes code for easy maintenance, and uses overloaded functions for readability. C++ is an excellent translation match for FORTRAN—and delivers more programming power!

■ Call today for more information and special product discounts!

COBALT BLUE

TM

COBALT BLUE, INC.
875 OLD ROSWELL ROAD, SUITE D-400
ROSWELL, GA 30076, USA
TEL (404) 518-1116, FAX (404) 640-1182

◆ Request 105 on Reader Service Card ◆

code section. This section consists of one or more separate segments, the number and type of which depend on the program and its memory model. Following the code is the initialized data section. Then comes the uninitialized data section, and finally the stack.


The Location Process

Once it has loaded a program into memory, the MS-DOS loader adds the code section's segment address to the segment value stored in each location requiring a fix up. The loader finds these locations by dereferencing each fix-up pointer in the header. MS-DOS sets the CPU's stack registers to *disp_stack_seg:sp* and calls the program at address *rel_cs_seg:ip*. (Note: For the sake of illustration, I use *disp_stack_seg*, *sp*, *rel_cs_seg*, and *ip* to represent values stored at specific offsets within the EXE header. By referring to fields of the same name in my *struct*, *EXE_HDR*, you can see where these values are stored in the header.) MS-DOS also provides some environment and header information to the loaded program through register contents.


Most, but not quite all of the information necessary to generate rommable absolute binary files already exists in the EXE file. The rest of the information must come from the segment data in the compiler's MAP file and from configuration information provided by the user in a loader configuration file.

Program Description

ROMLDR uses the linked EXE file and its MAP file to create a binary file that can be programmed into EPROMs.



Text Processing Tools for Your Applications



Cross Platform Cross Language

**UNIX | OS/2
WINDOWS | DOS**


'C' | PASCAL | BASIC | ASM

- Drop-in editing engines
- Spell Checking Engine
- Word Lists

Satisfy your text processing requirements with one of the **miniEd** Family of tools. Handle memo fields, pop-up windows, document entry, or any other formatted text requirement with a call to a **miniEd** function. 'C' source Included.

New - 'C' language Quality Assurance Toolkit

- Complete Test Suite coverage analyzer
- Traps for all types of memory corruption
- Available for MS-DOS - UNIX - OS/2



Strategic Software Designs, Inc.
6 S 235 Steeple Run Dr.-Ste 12B
Naperville, IL 60540
Phone: 708/778-6060
FAX: 708/778-6063
Mastercard VISA

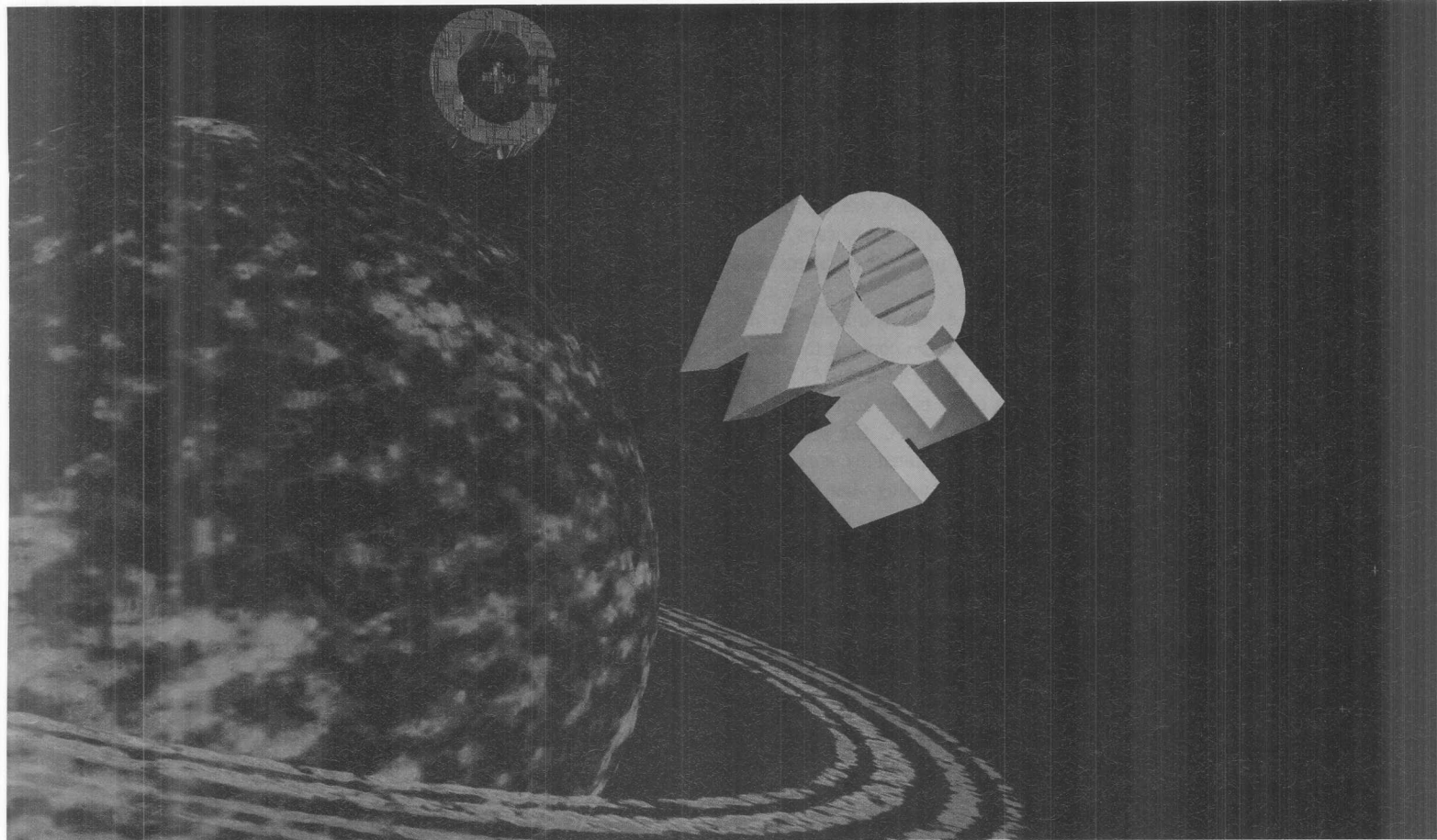
◆ Request 387 on Reader Service Card ◆

Listing 1 *continued*

```

/* ----- sort_table ----- */
/* sort header table */
void sort_table(void)
{
    qsort((void *)&header[filhdr->off_reloc],
          filhdr->num_reloc,4,cmp_ptr);
}
/* ----- read_block ----- */
size_t read_block(char far *segbuf,size_t segsz)
{
    if(fread(segbuf,1,segsz,exefile) != segsz)
        term_error(-7);
    return segsz;
}
/* ----- read_seg ----- */
long read_seg(int i)
{
    long segsize;
    segsize = mtable[i].haddr - mtable[i].addr;
    if(!segsize) return 0;
    segsize += mtable[i+1].addr - (mtable[i].haddr);
    if(segsize <= 0x8000)
    {
        read_block(seg_buffer,(size_t)segsize);
    } else {
        read_block(seg_buffer,0x8000);
        read_block(&seg_buffer[0x8000],
                  (size_t)(segsize-0x8000));
    }
    return segsize;
}
/* ----- write_block ----- */
size_t write_block(char far *segbuf,size_t segsz)
{
    if(fwrite(segbuf,1,segsz,binfile) != segsz)
        term_error(-8);
    return segsz;
}
/* ----- write_seg ----- */
long write_seg(long segsize)
{
    if(!segsize) return 0;
    if(segsize <= 0x8000)
    {
        write_block(seg_buffer,segsize);
    } else {
        write_block(seg_buffer,0x8000);
        write_block(&seg_buffer[0x8000],
                  (size_t)(segsize-0x8000));
    }
    return segsize;
}
/* ----- fix_seg ----- */
int fix_seg(int i)
{
    unsigned tmp,cseg,fixup;
    unsigned far * fixptr;
    cseg = (unsigned)(mtable[i].addr/16L);
    while(next_fix < filhdr->num_reloc)
    {
        if(ch_ptr[next_fix][1] > cseg) break;
        tmp = ch_ptr[next_fix][0]; /*offset into buffer*/
        fixptr = (unsigned far *) &seg_buffer[tmp];
        fixup = *fixptr;
        /* modify segment fixup according to type */
        if(fixup >= ramdata)
        {
            /* modify for ram */
            fixup -= ramdata;
            fixup += ramsadr;
        }
        else
        {
            /* handle as rom */
            fixup += romsadr;
        }
    }
}

```



OEW for C++ The Next Generation

A real CASE tool provides a true integration of your model, design, implementation and documentation. OEW delivers complete integration regardless of the size of your project or number of users.

OEW, "the Object Engineering Workbench", is a road tested working solution providing Object-Oriented Analysis, Design, Implementation and Documentation for C++ based applications. Utilizing a technically advanced approach, there is no longer a gap between model and code, they are simply two separate windowed views.

Automated Code Generation and Reverse Engineering are key functional components of the OEW development environment. OEW completely supports the certified AT & T C++ 3.0 standard. Import and export your entire program including header

and source files or those from commercial class libraries easily with the click of a mouse. Object-Engineering is a breeze with OEW because it supports the entire engineering cycle, "forward and reverse".

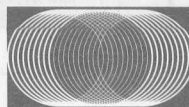
OEW is controlled by a powerful and intuitive user interface for several window environments, and includes a case sensitive on-line help system, a complete set of manuals including a getting started book and the "OEW meets Stroustrup's C++" training guide for professional C programmers.

If you are building software using the C++ Programming Language, reap the benefits enjoyed by OEW users around the world.

Call now tollfree for more information and ask about our **Free Small Project Version**.

✧ Request 223 on Reader Service Card ✧

**INNOVATIVE
SOFTWARE**
Innovative Software GmbH


OBJECT DESIGN
PARTNER

BSO/Tasking
(800) 458-8276 (in North America) or
(617) 320-9400, Fax (617) 320-9212

OEW International (Fax), Germany - Innovative Software ++49 (0) 69 236930, Australia/Asia - OSE ++61 3699 9234, BeNeLux - Protocols ++31 (0) 206 40 3341, UK - Silicon River ++44 (0) 81 316 7778

ROMLDR, an Embedded System Program Locator within the Borland IDE, I had to reduce the buffer's size significantly due to memory constraints.) A simple error routine, *term_error*, generates error messages for a variety of potential problems, and provides for program termination.

After allocating a buffer, *ROMLDR* reads the configuration file (CFG) specified on the command line. This file contains the names for the EXE, MAP, and BIN output files, EPROM and RAM hexadecimal load addresses, and the class name of the first RAM segment. Table 1 shows the CFG file format. CFG file parameters must be located on separate lines and separated by spaces. On each line, *ROMLDR* ignores any characters occurring after the list of required parameters.

Reading the MAP File

ROMLDR executes a *while* loop to read and process lines of text from the MAP file. (The MAP file used with *ROMLDR* should be the short version, which contains only the segment table.) *ROMLDR* extracts memory allocation class names, plus their starting and ending addresses, and stores them in an array of structures called *map_table*. *ROMLDR* performs a simple length check using configuration variable *class_loc* to determine if the current line contains segment information. *ROMLDR* expects the line to be in a fixed column format, with the class name occurring at offset *class_loc*. *ROMLDR* converts address values to long integers, and compares class names with *ram_class*, a configuration variable used to define the beginning

```
*fixptr = fixup;
next_fix++;
} /*end while */
return 0 ;
}

/* ----- hexcvt ----- */
/* do hex digits to unsigned long */
long hexcvt(char * num)
{
char *term;
long value;
value = strtoul(num,&term,16);
return value;
}

/* ----- gethdr ----- */
void gethdr(char *buf )
{
int i,j=32; /*index counter */
if(fread(&buf[0],1,32,exefile) < 32)
term_error(-6);
/* have filhdr contents so get size of full header */
if(fread(&buf[j],16,filhdr->hdr_siz-2,exefile)
< filhdr->hdr_siz-2) term_error(-6);
(unsigned *)ch_ptr =
(unsigned *)(&buf[filhdr->off_reloc]);
/* get address of relocation table - ch_ptr [n][m]
m - 0 offset, 1 - seg, n relocation # */
}

/*----- config -----*/
/* get configuration data */
int config(char *cfgfile)
{
FILE *cfg;
char buf[80];
if((cfg = fopen(cfgfile,"r"))==NULL) term_error(-1);
if(fgets(buf,80,cfg) == NULL) term_error(-2);
if(sscanf(buf,"%s %s %s",&mfile,
&efile, &bfile) != 3) term_error(-2);
/* Now try to open input file 1 */
if((mapfile=fopen(mfile,"r"))==NULL) term_error(-3);
if((exefile=fopen(efile,"rb"))==NULL) term_error(-4);
if((binfile=fopen(bfile,"wb"))==NULL) term_error(-5);
if(fgets(buf,80,cfg) == NULL) term_error(-7);
if(sscanf(buf,"%4x %4x",&romsadr,
&ramsadr) != 2) term_error(-7);
if(fgets(buf,80,cfg) == NULL) term_error(-7);
if(sscanf(buf,"%s",&ram_class) != 1) term_error(-7);
return 0;
}

/* error handler */
char *errlist[10] = {
"Memory Allocation Error",
"No configuration file, USAGE:romldr cfgfile.cfg",
"Configuration file error - File names", //-2
"Map File open error", //-3
"Exe File open error", //-4
"Bin File open error", //-5
"Error reading header", //-6
"Error reading exe file", //-7
"Error writing bin file", //-8
""
};

void term_error(int errnum)
{
errnum = abs(errnum);
if(errnum >= 9) exit(-1);
printf("%s\n",errlist[errnum]);
farfree(seg_buffer);
exit(errnum);
}

/* End of File */
```

The SnooperTM

Ethernet Protocol Analyzer

Become a Networking Expert with this PC-Based Analyzer for NetWare and LAN Manager LANs.

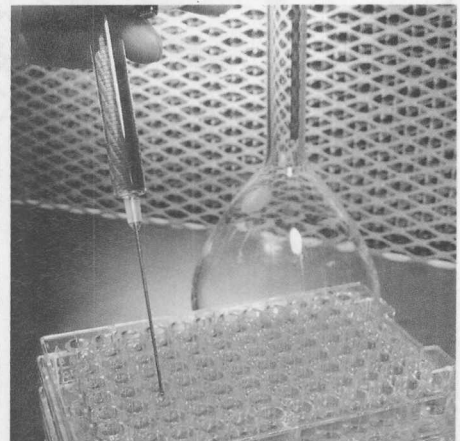
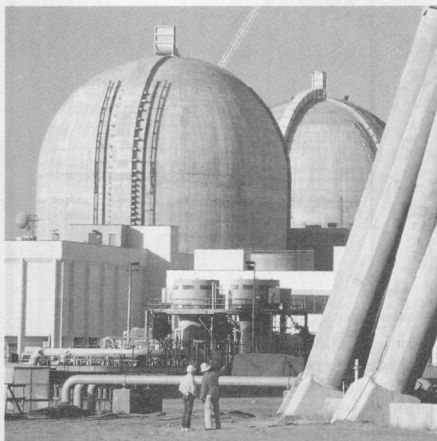
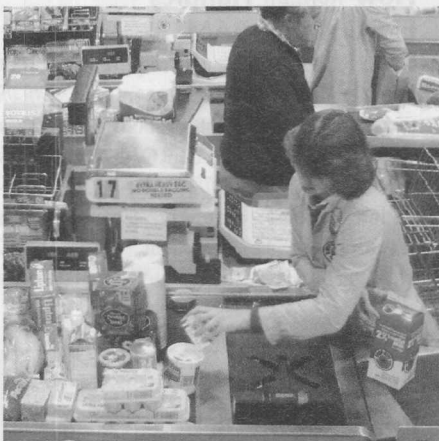
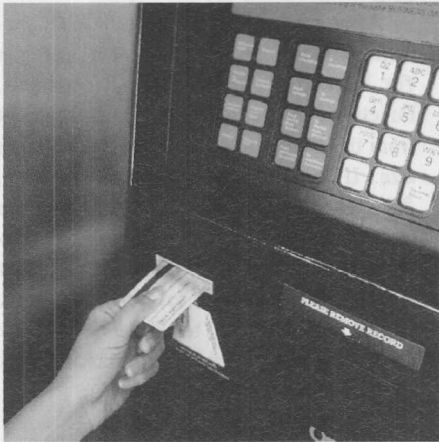
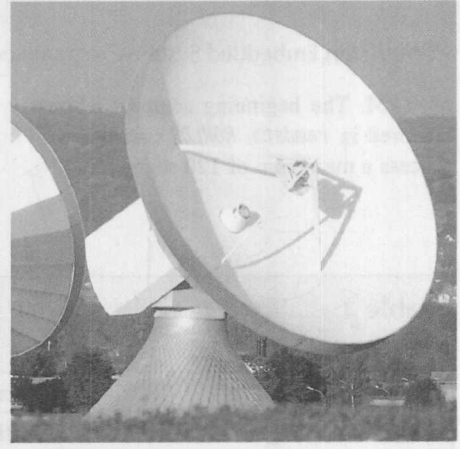
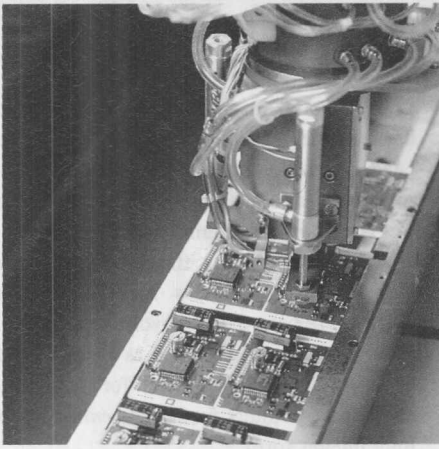
Captures, decodes into multiple layers, and displays Ethernet traffic, and lets you actually add your own protocol interpreters. With source, only \$350.

**GENERAL
SOFTWARETM**
P.O. Box 2571 Redmond, WA 98073

Tel. (206) 391-4285
Fax. (206) 746-4655
The Protocol Experts

Copyright (C) 1992 General Software, Inc. General Software, the GS logo, and The Snooper are trademarks of General Software. Other marks property of their owners.

◆ Request 321 on Reader Service Card ◆



SOME APPLICATIONS NEED A *REAL* REALTIME OPERATING SYSTEM

Whether you're monitoring a nuclear reactor or handling credit-card transactions, you need realtime performance you can count on around the clock. The kind of performance QNX® has delivered for well over a decade. QNX sustains a host of successful mission-critical applications in a wide range of industries. From POS to medical instrumentation, from SCADA to voice mail, thousands of VARs and OEMs rely on QNX for their realtime solutions. It's easy to see why. QNX is a microkernel OS combining realtime executive-class speed with a rich, self-hosted development environment. You won't have to waste time on cross-

development, so you'll see faster time-to-market and easier maintenance for your applications. QNX is remarkably flexible. You can easily strip it down to an embedded system or build it up to a vast network serving hundreds of CPUs. QNX follows Open Systems standards like POSIX and TCP/IP, so all your applications become all the more portable and interoperable. And you can run most popular DOS packages—even Microsoft® Windows™ 3.1 in standard mode under QNX. So if you're looking for a time-tested foundation for your mission-critical applications, it's time for a *real* realtime solution. It's time for QNX.



WE WORK IN REAL TIME.™

1-800-363-9001

(EXT. 102)

QNX SOFTWARE SYSTEMS LTD. 175 TERENCE MATTHEWS CRESCENT, KANATA, ONTARIO, CANADA K2M 1W8 TEL: 613-591-0931 • FAX: 613-591-3579

© QNX Software Systems Ltd. 1993 QNX is a registered trademark of QNX Software Systems Ltd. Microsoft is a registered trademark and Windows is a trademark of Microsoft Corporation.

◆ Request 224 on Reader Service Card ◆

of RAM. The beginning segment address is stored in *ramdata*. ROMLDR currently will process a maximum of 120 segments.

Processing the Header

Once ROMLDR has acquired the MAP file, it reads the EXE header portion via function *gethdr*. This function first reads

Table 1 CFG file format

Mapfile Exefile Binfile	// filenames with no path
ROM_Address RAM_address	// 4 digit hex addresses [0...9,a...f]
Ram_Class	// name for first RAM segment

One source is all you really need.

New Visual C++
MFC support!

The news today is the Wind/U™ portability toolkit! Wind/U enables your Microsoft Windows application to run as a native UNIX/Motif application. You can use your Microsoft Windows source base to rapidly penetrate the fast growing workstation marketplace, or use Wind/U as a dual development environment for both Windows and Motif.

The advantage of Wind/U is that the toolkit uses the complete set of the Microsoft Windows application programming interface (API) facilities under Motif. You only need to recompile your Windows application in the UNIX environment and link to the Wind/U Library. Wind/U maps the Windows function calls to native Motif and X calls, allowing the Windows application to execute in the native UNIX environment.

Wind/U ported applications maintain identical functionality between Windows and Motif versions by providing features such as on-line help, PostScript and PCL printing support, DDE, MDI, and Common Dialogs.

Now you can cost-effectively run your applications on Sun, IBM, Hewlett-Packard, and DEC workstations, while focusing your programming efforts on the popular Windows API.

For more information call (203) 438-6969, email to info@bristol.com or fax to (203) 438-5013.



Bristol Technology Inc.

Bringing the Best of Microsoft Windows to X and Motif

Wind/U is a trademark of Bristol Technology Inc. PCL is a registered trademark of Hewlett-Packard. Sun is a trademark of Sun Microsystems, Inc. IBM is a registered trademark of International Business Machines, Inc. UNIX is a registered trademark of UNIX Systems Laboratories, Inc. Microsoft is a registered trademark and Windows is a trademark of Microsoft Corporation. Motif is a registered trademark of the Open Software Foundation. PostScript is a registered trademark of Adobe Systems, Inc.

in 32 bytes of the header to determine the header's size and then loads the rest of the header. This function stores header information in an array named *header* which can contain a maximum of 10,000 bytes.

ROMLDR then sorts the fix-up pointers into ascending address order. Function *sort_table* uses *qsort* to sort the pointers. Function *cmp_ptr*, supplied as an argument to *qsort*, compares values for *qsort* by converting pointers from *segment:offset* to long integer form.

After sorting the pointers, ROMLDR performs fix ups on each segment, using a *for* loop to iterate through all segments. Function *read_seg* reads each segment from the EXE file and returns the segment size. If the segment length is non-zero, ROMLDR calls function *fix_seg* to run through any fix ups needed for the segment and then calls *write_seg* to output the processed segment to the BIN file. (A logical enhancement to ROMLDR would be to output the segments in a standard hex format, such as Intel hex format.)

How ROMLDR Handles EPROM and RAM

EPROM segments require different modifications than RAM segments. ROMLDR treats all segments at or below the class named *ENDCODE* as EPROM and treats those above *ENDCODE* as RAM. The location process currently terminates on reaching the last segment, the *STACK* class. (The BIN file, however, needs only

Listing 2 Sample program that produces segment classes

```
DEMO.C
/* Test Program for Rom Loader */
/* Large model with some far data*/
#include <conio.h>
#include <dos.h>

char msg[15] = "Hello World\n\r";
int locint;
int directvideo=1; /*BC direct to hw*/
int far test;
int far test2=0x55aa;
int far * tptr = &test;

void main(void)
{
    int far * ptr2 = &test;
    test = 0x1111;
    locint = test+1;
    cputs(msg);
}
/* End of File */
```

to contain code and data segments up to the last initialized data value.)

The *ENDCODE* class name is special for another reason. ROM based systems typically must transfer initialized data from EPROM to RAM. Therefore, *ROMLDR* will modify all references to data segments to refer to the RAM locations and not to the initial values located in the EPROM. (The location for the initial values must be used in the startup code so that they can be transferred to RAM.) Segment *ENDCODE* is used for this purpose. I make the *ENDCODE* segment's length less than 16 bytes, locate it on a paragraph boundary, and ensure that the beginning data segment is also aligned by paragraph. As a result, the beginning ROM location for initialized data becomes *ENDCODE*+1. Since *ENDCODE*'s address is less than the beginning of the RAM segment, it will refer to the ROM address just below the initialized data values.

ROMLDR modifies EPROM data by adding the configuration file's EPROM segment address, stored in *romsadr*, to the code's existing segment value. (A more sophisticated version of the program could offer the option of several user-defined addresses and the names of the classes that would reside in each.)

ROMLDR modifies RAM segments by first subtracting out the value of the first RAM segment and then adding the configuration file's RAM segment location value. This method allows the RAM locations to begin at the configuration-defined starting value. The subtraction was not necessary for code segments since they began with a zero segment value.

Example Code

Listing 2 is *DEMO.C*, a typical "Hello World" program with some added items to provide examples of values for several segment classes. Listing 3, *DEMO.MAP*, is the map file generated for *DEMO.C* using the example startup code in Listing 4 instead

Coding for Embedded Applications

Writing embedded systems applications requires special efforts on the part of the programmer, because of how these applications differ from non-embedded applications. First, an embedded program that crashes can cause damage or injury, while a non-embedded program (e.g. a word processor) may cause only a certain amount of user frustration. Embedded systems must handle the conceivable problems in stride and effectively recover from the inconceivable without help from users.

Second, there are several special aspects to embedded code. Many embedded systems run standalone programs without the support of complete operating systems such as MS-DOS, so these programs must take control of relevant interrupt vectors (including error conditions such as divide by 0 and the Non-Maskable Interrupt). For these standalone programs, both the

hardware and the program may require setup code. Programs compiled to run under MS-DOS contain startup and termination code to accept control from and return control to the operating system. Standard library functions that don't access MS-DOS under normal conditions may contain error-handling code that does access MS-DOS or that would terminate the program in an unacceptable fashion within an embedded system. In addition, some programs may attempt to access non-disk MS-DOS functions, such as those used to modify interrupt vectors. If you port such programs to an embedded system, you must provide code to perform equivalent functions within the embedded system operating environment. An embedded program must start (or restart) in a known state. Since the program cannot obtain state information from a command line, it must find initial state data in EPROM. □

AMXTM The Real-Time Multitasking Kernel

680x0, 683xx
80x86/88 real mode
80386 protected mode
i960[®] family
R3000, LR330x0
Z80, HD64180

Features

- Full-featured, compact ROMable kernel with fast interrupt response
- Preemptive, priority based task scheduler with optional time slicing
- Mailbox, semaphore, resource, event, list, buffer and memory managers
- Configuration Builder utility eases system construction
- InSight[™] Debug Tool is available to view system internals and gather task execution statistics
- Supports inexpensive PC-hosted development tools
- Comprehensive, crystal clear documentation
- No-hidden-charges site license
- Source code included
- Reliability field-proven since 1980

Count on KADAK.
Setting real-time standards since 1978.

For a **free** Demo Disk
and your copy of our excellent AMX
product description, contact us today.

Phone: (604) 734-2796
Fax: (604) 734-8114



KADAK Products Ltd.
206 - 1847 West Broadway
Vancouver, BC, Canada V6J 1Y5

AMX is a trademark of KADAK Products Ltd.
All trademarked names are the property of their
respective owners.


```

DEMO.MAP
Start Stop Length Name Class

00000H 00000H 00000H ROMSTART_BEG CODE
00000H 00E8FH 00E90H _TEXT CODE
00E90H 00EC9H 0003AH DEMO_TEXT CODE
00ED0H 00ED5H 00006H ENDCODE ENDCODE
00EE0H 00EE0H 00000H IDATA_BEG IDATA_BEG
00EE0H 00EE0H 00000H FAR_DATA FAR_DATA
00EE0H 00EE3H 00004H DEMO5_DATA FAR_DATA
00EF0H 00FB9H 000CAH _DATA DATA
00FBAH 00FBBH 00002H _CVTSEG DATA
00FBCH 00FC1H 00006H _SCNSEG DATA
00FC2H 00FC2H 00000H CONST CONST
00FC2H 00FC7H 00006H _INIT_ INITDATA
00FC8H 00FC8H 00000H _INITEND_ INITDATA
00FC8H 00FC8H 00000H _EXIT_ EXITDATA
00FC8H 00FC8H 00000H _EXITEND_ EXITDATA
00FD0H 00FD0H 00000H IDATA_END IDATA_END
00FD0H 00FD0H 00000H UDATA_BEG UDATA_BEG
00FD0H 00FD1H 00002H _BSS BSS
00FD2H 00FD2H 00000H _BSSEND BSSEND
00FE0H 02FDFH 02000H STACK STACK
02FE0H 02FE0H 00000H UDATA_END UDATA_END

```

Program entry point at 0000:0000

Listing 4 An example of startup code

```

NAME ROMSTART_TEXT
;
; example startup code for
; imbedded systems use
;
; segment classes
; High addr (rom)
; 'BOOT'
; ""
; 'CODE' _TEXT segment
;
; ram
;
; 'STACK' UDATA_END segment
; 'BSS' IDATA_END and UDATA_BEG segments
; 'CONST'
; 'DATA' IDATA_BEG segment
;
; Rename object file output to c0x.obj
; where x = S,C,M,L, or H
; and locate in project subdirectory
; Memory model selection set to 1 all others 0
;
SMALLM EQU 0
COMPACTM EQU 0
MEDIUMM EQU 0
LARGEM EQU 1
HUGEM EQU 0

STACK_SIZE EQU 1000H ;set desired stack size
_acrtused equ 1 ;satisfy external reference

PUBLIC _acrtused

DGROUP GROUP IDATA_BEG, _DATA, CONST, IDATA_END, \
UDATA_BEG, _BSS, STACK, UDATA_END

```

```

; this segment marks beginning of rom code
ROMSTART_BEG SEGMENT BYTE 'CODE'
ROMSTART_BEG ENDS

```

```

IF SMALL OR COMPACTM
_TEXT SEGMENT BYTE PUBLIC 'CODE'
EXTRN _main:NEAR ;main program
ASSUME CS:_TEXT

```

```

ENDIF
IF MEDIUM OR LARGEM OR HUGEM
EXTRN _main:FAR ;main c program
_TEXT SEGMENT BYTE PUBLIC 'CODE'
ASSUME CS:_TEXT
ENDIF

```

```

ASSUME DS:DGROUP, SS:DGROUP
PUBLIC start

```

```

start PROC NEAR
cli
cld
;*****
;
; do hardware initialization and ram check
;
;*****

```

```

PUBLIC init_ram

```

```

init_ram:
;*****
; transfer initialize data from rom to ram
;

```

```

MOV BX, SEG IDATA_BEG
MOV AX, SEG IDATA_END ;data to init.
sub ax,bx
mov cl,3
shl ax,cl
mov cx,ax
jcxz no_init_data

```

```

;address of frame # in rom
mov ax, seg ENDCODE ;needed for jbrmldr
inc ax ;ram init values begin at
mov ds,ax ;segment ENDCODE + 1
mov si,0

```

```

;address of frame # in ram
mov ax, seg IDATA_BEG
mov es,ax
mov di,0

```

```

; initialize data and const segments
rep movsw ;word transfer
no_init_data:

```

```

mov bx, seg UDATA_BEG ;clear bss data
mov ax, seg UDATA_END
sub ax, bx
mov cl,3
shl ax,cl
mov cx,ax
jcxz no_zerodata
mov es,bx
mov di,0
mov ax,0

```

```

rep stosw
PUBLIC no_zerodata

```

```

no_zerodata:
mov ax,DGROUP ; set up stack
mov ds,ax
mov ss,ax
mov sp,OFFSET DGROUP:STACK_TOP

```

of the standard Borland startup code. Note that the `_INIT_` segment contains some values which are addresses of library initialization routines that should be called in order of priority. The example startup code does not yet include this section or the inter-

rupt vector initialization section. Examples of these can be found in your compiler's startup code. The sidebar "Startup Code for Embedded Systems" provides a discussion of startup code requirements.

Startup Code for Embedded Systems

Startup code consists of code to setup and to terminate an application program. Startup code is typically written in assembly language and the source code is sometimes provided with the C compiler. Replacement libraries often include replacement startup code that can be used when the normal C compiler library functions are not used.

Startup code for EPROM-based systems must provide additional functionality which depends on the specifics of the system.

Embedded system startup code usually performs the following steps:

1. Establish order of segment classes.
2. Set up the program stack.
3. Transfer initialized values to RAM from EPROM.
4. Zero uninitialized RAM values.
5. Set up error interrupt vectors.
6. Call necessary library initialization routines.
7. Call the application.

The startup code must also provide the following components:

8. Error shut-down code to terminate or restart.
9. Exit shut-down code to terminate or restart.
10. Any error and miscellaneous routines.
11. Initialization of application before calling main function.
12. A reset vector for standalone applications.

There are two approaches to creating a custom startup module. You can start with the original library or compiler code and modify it to handle the added embedded requirements, or you can start with a minimal embedded system startup module and extend it as required.

Custom startup code may not provide all features described in a compiler's documentation. `Environment`, `argv` and `argc`, and even common option variables may not be implemented. For example, Borland C allows you to set stack size by initializing a far variable, `_stklen`. This variable is created by the Borland startup code and may be replaced in custom startup code by a simple stack-size definition.

Startup code defines segments and segment classes and their order. While compilers often allow these names to be

changed for modules, the compiled libraries are still expecting certain module names to exist. Warnings abound over changing the basic segment order so extreme care should be used if that appears to be necessary. The addition of segments and classes is much less critical. As shown in Listing 4, you can even add them virtually to make the MAP more descriptive.

Listing 4 is a simple example of startup code. I provide it as a template rather than a complete example since some portions are compiler dependent. You may still need to add several code sections for an application. Refer to your compiler's startup code module for specific details. □

The Version Control Solution



SourceSafe™

The Only Object-Oriented
VCS for All Platforms

"Revolutionary, my dear Watson."

🏆 **Winner:** Microsoft Systems Journal's competitive review

🏆 **Winner:** Computer Language's Productivity Award

✓ **Endorsed By:** Novell, Intel, Oracle, Motorola, American Airlines...

SourceSafe™ is:

- 🔗 Project-oriented version control
- 🔗 A "FileManager" for your source code
- 🔗 Cross-platform: DOS, WIN, OS/2, NT, UNIX, MAC ...
- 🔗 Incredibly easy to use

SourceSafe™ will:

- 🔗 Install in 10 minutes
- 🔗 Save you a minimum of 5 hours a week
- 🔗 Track entire programs and projects
- 🔗 Never lose a file

"Ingenious!"

"Fiendishly clever!"

Call Now for a risk-free look at **SourceSafe™** and register to win 1 of 10 Microsolutions CD-ROM players (\$500 value).
Limited to the first 1000 callers.

1 • 800 • 364 • 5467 or fax 1 • 919 • 848 • 0307

◆ Request 193 on Reader Service Card ◆

ROMLDR Versus Commercial Products

ROMLDR has several shortcomings when compared to commercial locator packages. When you use ROMLDR you must provide startup code for your application; commercial products usually provide the basic startup code required as well as code solutions for a variety of problems which must be overcome in various embedded configurations. ROMLDR does not provide debugging support, but commercial products usually provide some capabilities for the debugging of application programs. Finally, vendors of commercial locator packages often provide technical support; when you use a non-commercial package such as ROMLDR you must solve all problems on your own.

Conclusions

While ROMLDR is intended primarily as a learning tool and an introduction to embedded systems, it can prove useful for some low-end applications. ROMLDR should be able to handle straightforward applications where there is one EPROM and one RAM memory space. It can easily be modified for more complex configurations, especially those which have specific fixed requirements.

Embedded systems often monitor and control equipment other than normal computer peripherals. Embedded systems programmers must be extra cautious, since bugs in their programs can place property and lives at risk. In this situation, there is no substitute for understanding both the application and the tools. Understanding how ROMLDR works may give you insight into how more complex systems operate. □

Listing 4 continued

```

;*****
; add code to:
; 1) initialize INITDATA Functions (see c0.asm)
; 2) capture interrupt vectors 0-4
;*****
sti ;enable interrupts
call _main ;enter main program
;*****
; add code to:
; 1) shut down EXITDATA Functions
; 2) handle shutdown errors
; 3) prepare to restart
;*****
    jmp start
start ENDP
_abort PROC DIST
    PUBLIC _abort
;handle error abort
    jmp start
_abort ENDP
IF SMALLM OR COMPACTM
    _TEXT ENDS
ENDIF
IF MEDIUMM OR LARGEM OR HUGEM
    _TEXT ENDS
ENDIF

; segment marks end of rom code
;make it non zero, length < 16 bytes
ENDCODE SEGMENT PARA PUBLIC 'ENDCODE'
    db "ENDROM" ;seg ENCODE+1 = begin ram
ENDCODE ENDS
;*****

;beginning of dgroup and initialized data in ram

IDATA_BEG SEGMENT PARA PUBLIC 'IDATA_BEG'
IDATA_BEG ENDS

_FARDATA SEGMENT PARA PUBLIC 'FAR_DATA'
_FARDATA ENDS

_DATA SEGMENT PARA PUBLIC 'DATA'
_DATA ENDS

_CVTSEG SEGMENT WORD PUBLIC 'DATA'
    PUBLIC __RealCvtVector
    __RealCvtVector label word
_CVTSEG ENDS
_SCNSEG SEGMENT WORD PUBLIC 'DATA'
_SCNSEG ENDS

```

```

CONST SEGMENT WORD PUBLIC 'CONST'
CONST ENDS

_INIT SEGMENT WORD PUBLIC 'INITDATA'
_INIT ENDS
_INITEND SEGMENT WORD PUBLIC 'INITDATA'
_INITEND ENDS

_EXIT SEGMENT WORD PUBLIC 'EXITDATA'
_EXIT ENDS

_EXITEND SEGMENT WORD PUBLIC 'EXITDATA'
_EXITEND ENDS

IDATA_END SEGMENT PARA PUBLIC 'IDATA_END'
IDATA_END ENDS

; end of initialized data

UDATA_BEG SEGMENT WORD PUBLIC 'UDATA_BEG'
UDATA_BEG ENDS

_BSSSEGMENT WORD PUBLIC 'BSS'
_BSS ENDS
_BSSSEND SEGMENT BYTE PUBLIC 'BSSSEND'
_BSSSEND ENDS

STACK SEGMENT PARA STACK 'STACK'
    DW STACK_SIZE DUP (?)

STACK_TOP LABEL WORD
STACK ENDS

; end of initialized segment in ram
UDATA_END SEGMENT WORD PUBLIC 'UDATA_END'
UDATA_END ENDS
; bootstrap address for powerup reset
; far jump to program beginning
; may have to be manually placed in EPROM
;BOOTSTRAP SEGMENT AT 0FFFFH
;    JMP FAR PTR start
;
;BOOTSTRAP ENDS
    end start

; End of File

```

A Fuzzy-Logic Torque Servo

Jack J. McCauley

Background

The developer of a control system desires a well-behaved system that is stable throughout the operating spectrum. In the process of designing and debugging the controller, operating regions that exhibit oscillatory or unstable operation are avoided. Non-linearities are typically modeled using a series of discrete equations developed for the system.

A PID or Proportional Integral Differential controller is an example of a closed-loop system. Implemented discretely, it would compute at z intervals:

$$T(z) = T(z-1) + K_k * (E(z) - E(z-1)) \\ + K_i * E(z) + K_d * E(z-2) * (1 + E(z-1))$$

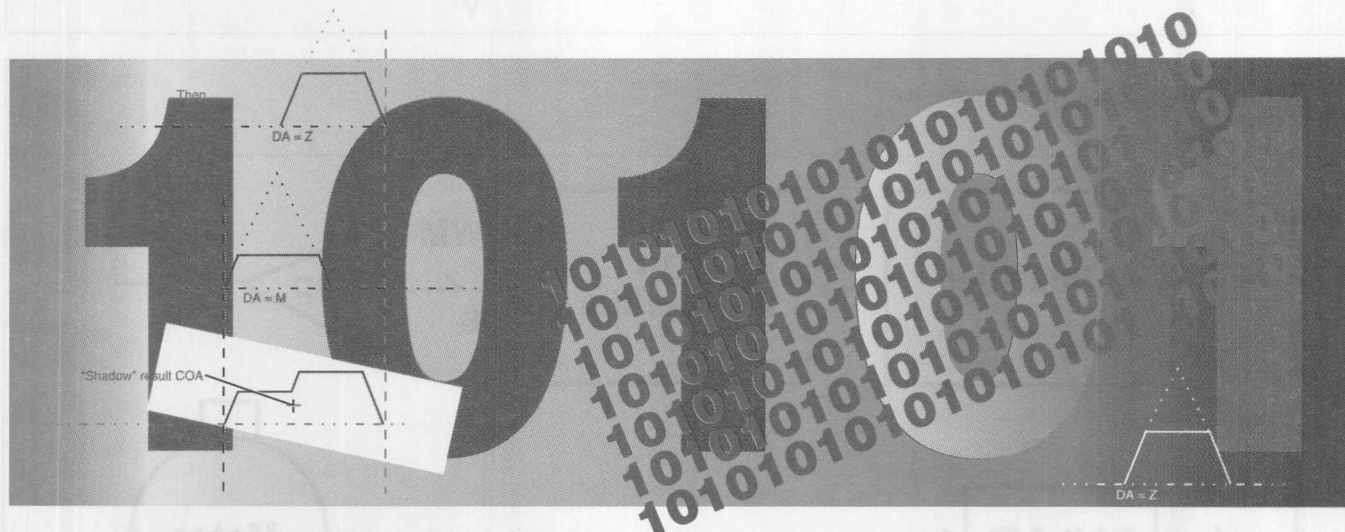
In this linear example, the z sampled periods for the controller represent discrete time samples and K_k , K_i , and K_d are constants. A non-linear system designer might attempt to modify K_k , K_i , and K_d as a function of input $E(z)$. This of course would require $K_k(z)$, $K_i(z)$, and $K_d(z)$ (and possibly $K_k(z-1, z-2, \dots, z-n)$ etc). The designer would compute stability and phase margin in a continuous system and convert these to the discrete world using a Discrete Transform.

After the analysis the designer would proceed to code $T(z)$ into assembler or C and port it to a target. This article describes how to implement a simple *fuzzy-logic* based servo controller in the C programming language. C language portability allows the controller to run in either a micro-controller based environment or on a PC.

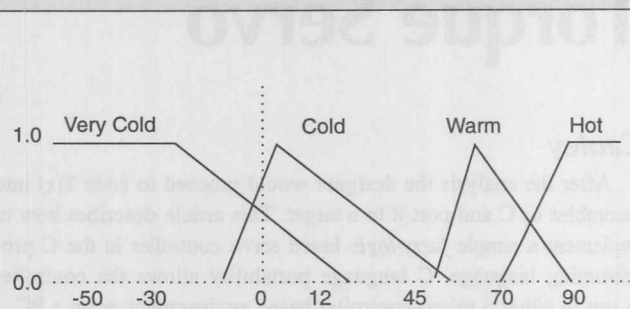
Escaping Brittleness

Brittleness is the bane of the control-system designer. A brittle system is one which "breaks" easily. For example, in the coding of $T(z)$, we defined the output $T(z)$ to be an eight-bit *unsigned char* that follows the eight-bit radix of our A/D converter. Suppose that several of the terms of $T(z)$ were discretely nine or ten bits. The result could be truncated and thus produce incorrect output when $T(z)$ was cast to an *unsigned char*. We must check overflow after every operation of $T(z)$ and perform corrective logical operations.

Besides the implementation issues, PID and PI suffer from some serious real-world problems. In controls, non-linearities are the rule. Factors such as friction and temperature affect the behavior of systems. What appears to be stable under one set of conditions yields poor performance under another.



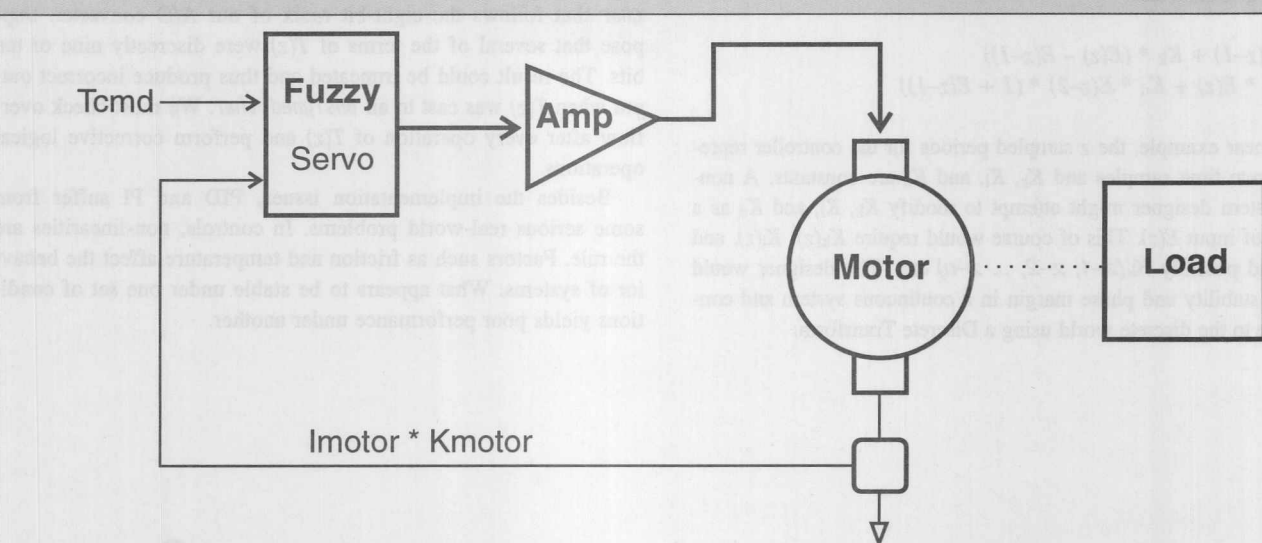
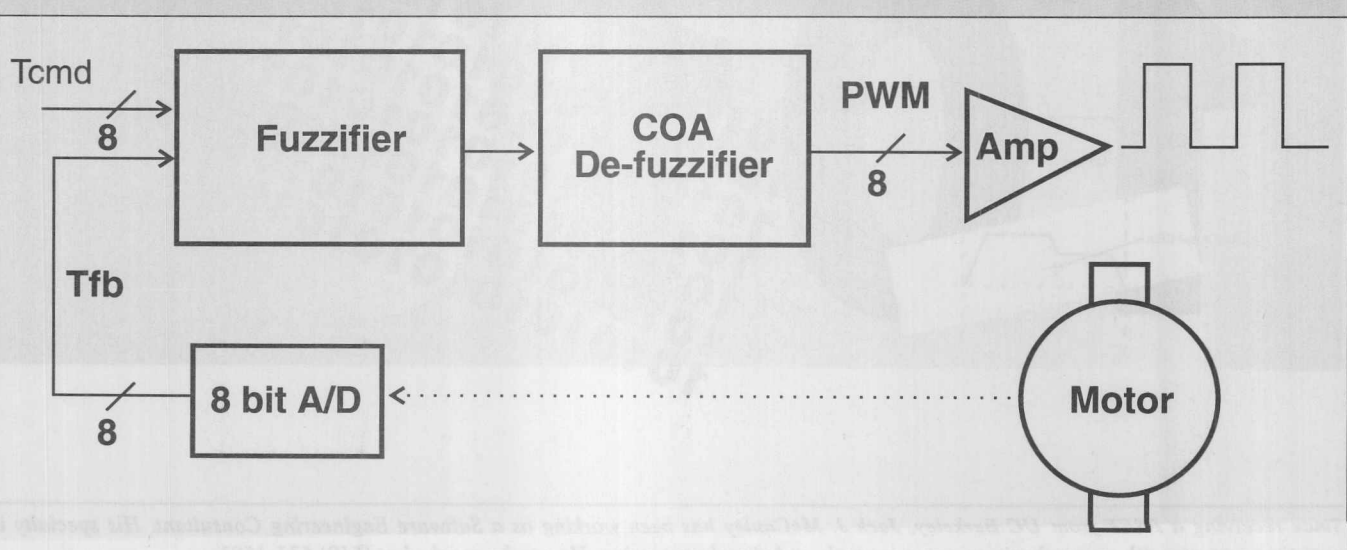
Since receiving a BSEE from UC Berkeley, Jack J. McCauley has been working as a Software Engineering Consultant. His specialty is real-time systems with an emphasis on servo controls and signal processing. He can be reached at (510) 531-1581.

Figure 1 Fuzzy set for temperature

There are techniques for dealing with changes. One possibility is to *sliding-mode* modify the PI gain parameters. Experience with this approach has shown that it works very well so long as the inputs are bounded properly. Current thinking, however, indicates that the amount of time invested in developing this approach will not be extracted in performance. What most people do to deal with non-linearities is to add a bunch of testing and branching code to deal with the caveats. In most PID, PI closed-loop systems, it ends up that the majority of the code is dedicated to dealing with these anomalies.

Fuzzy Logic

Fuzzy logic holds promise as a means of handling control-system non-linearities. It provides a unique way of looking at control

Figure 2 Feedback configuration of a typical torque servo**Figure 3** Fuzzy plant for torque servo

problems. The control problem is described as a rule base, with a rule input matrix $\mu(T)$ and a corresponding output function for that rule. When few rules are to be evaluated, simple linguistic rules can be substituted instead.

One might imagine a fuzzy set $\mu(T)$ that describes the outside temperature as "very cold," "cold," "warm," and "hot." The four members of $\mu(T)$ are linguistic operators that represent the human inferred description of the space representation of each member of $\mu(T)$. The membership function is typically a trapezoid that represents a degree of membership. The degree of membership is a real number between 0.0 and 1.0 with a degree of 1.0 meaning full membership and a degree of 0.0 indicating no membership. (See Figure 1.)

Suppose the outside temperature is -50 F. Then according to $\mu(T)$ the outside temperature will lie in the domain of "very cold." If the temperature is 12 F, the temperature might not be so much "very cold" as it is just "cold." Membership functions almost always overlap in the domain of $\mu(T)$. Fuzzy logic provides a means of dealing with overlapping domains and also domains of overlapping output sets. The math allows us to weigh the cumulative effect of all rules to generate a *crisp* output. A crisp output is also called a *de-fuzzified* output.

The typical fuzzy plant (controller) consists of one or more input fuzzy sets, a rule base, and an output fuzzy set. The input sets fuzzify using the rule base, and the output set de-fuzzifies from the inputs, rules, and the output de-fuzzifier function.

An Application

This two-input controller is a torque regulator on a DC motor. The controller topology could easily be extended to any two-input, single-output fuzzy controller.

We desire to regulate the torque applied to the shaft of a permanent-magnet DC brush motor. The torque T_{motor} applied to the shaft is a linear function of armature current I_{motor} and (to a lesser extent which will be ignored here) motor temperature. Armature current is controlled by a Pulse Width Modulated (PWM) amplifier which varies the duty cycle DA from 0 to 100 per cent as a linear function of input voltage.

If a positive voltage is applied to the motor terminals, a positive torque will be applied to the motor shaft. If a negative voltage is applied, a negative torque will be applied to the motor shaft. The job of the servo is to regulate the torque on the motor shaft under varying load conditions. The servo applies an adjustment to DA based on the commanded torque input, $T_{\text{cmd}}(z)$, and the feedback torque input:

$$T_{\text{arm}} = I_{\text{motor}}(z) * K_{\text{motor}}(z)$$

sensed on the motor shaft. (See Figure 2.)

The eight-bit PWM applies a positive voltage to the terminals if the PWM value is between 128 and 255. A negative voltage is applied if the value is between 0 and 126. An eight-bit A/D converter and pre-amp sense the feedback torque. The granularity of the feedback torque is determined by the radix of an eight-bit A/D converter. An eight-bit D/A (PWM) assures coherency between conversion radices on the input. (See Figure 3.)

C/TASKER

The multitasking library toolkit for C

Run your existing C functions with few or no changes as concurrent threads! Only C/Tasker lets you work in pure, standard C and freely use any standard library function or DOS call. Build high-performance multi-line serial apps. (e.g., one thread per channel). Features: semaphores, event, message queues, critical sections and much more. No re-entrancy problems with DOS or any C function. Ready-to-compile on-disk sample source code. See more than 1,000 threads run with our free demo (source code included). Run Turbo Vision as a thread and spawn more threads from within it! Real-time support: ISRs may resume suspended threads. Blindingly fast scheduler, designed for use in commercial-quality applications. Compatible with Borland, Microsoft and Symantec C, Turbo Debugger, Codeview, BGI, floating point emulators, 80x87.

\$249

C/TASKER PLUS

One step ahead of the crowd

C/Tasker PLUS includes every feature of C/Tasker and adds full source code and the Task Driver Toolkit, an exclusive concept for the creation of new multitasking facilities (installable services). With Task Drivers, you work with a solid API to build your own multitasking primitives (e.g. IPX/NETBIOS network communication functions that your threads may use to establish non-blocking, concurrent peer-to-peer communications over the network). An interrupt-driven serial communication Task Driver is included and may be used as is, or as a basis for further development.

\$995

(only \$749 when upgrading from C/Tasker)

TURBOWORD

The word processor toolkit for C

TurboWord is a full-blown word processor / text editor library that may be linked to your programs. Customize it anyway you like; the proposed command dispatcher is WordStar-like. Features: dynamic wordwrap; horizontal scroll; fast background printing; search/replace with options; hyphenation; block ops; undo; bold/italics/underline; dotted commands; and more. Extremely compact and fast. Of course, as a C/Tasker-compatible library, it may run concurrently with your threads. May work in a window without disturbing the rest of the screen. Full source code is included.

\$69

DEL

Data Encryption for C, C++, Clipper

DEL (Data Encryption Library) is a high-speed toolkit that lets you encrypt/decrypt data in C, C++ and Clipper. It includes: a software-only, cipher-feedback DES engine; a high-performance FastDES mode; an ultra-fast Turbo mode; and blazing-speed CRC-16 and CRC-32 routines. The library is DOS and Windows compatible, and is supplied in both .LIB and .DLL formats. Compatible with Borland, Microsoft and Symantec C/C++ and Clipper. Full source code available.

\$119

(for source, add \$100)



For info and technical support, contact:
North and South America: ESC, 913-832-2070;
Fax: 913-832-8787; CompuServe: 71141,3624
Elsewhere: Microware sas, Italy, +39-6-3332744;
Fax: +39-6-3336465; e-mail: mc6388@mcclink.it

◆ Request 204 on Reader Service Card ◆

Fuzzy Controller for Torque

The fuzzy controller is implemented in C on an inexpensive high-integration micro-controller. A simulation program executes on a PC-AT, to assist debugging of the fuzzy servo prior to porting it to the micro-controller.

In this system the fuzzy plant is a two-input, single-output controller. Three fuzzy sets exist with the input sets occupying two dimensions of the rule base and the output set occupying the action to be performed on each rule, which is the combination pointed to by the input fuzzy set membership functions.

The first input fuzzy variable T_{error} is the error between the setpoint or command torque at time t , and the feedback torque or:

$$T_{\text{error}}(z) = T_{\text{cmd}}(z) - T_{\text{arm}}(z)$$

The second input dT_{error}/dt is the "rate of change of error" or how quickly the feedback torque at time t is changing with respect to the previous sample at time $t-1$:

$$dT_{\text{err}}(z)/dt = T_{\text{arm}}(z) - T_{\text{arm}}(z-1)$$

The output function $\mu(DA)$ determines the action to be performed upon evaluation of the rules. For example, if the instantaneous feedback torque is at the setpoint:

$$T_{\text{cmd}}(z) \sim T_{\text{arm}}(z)$$

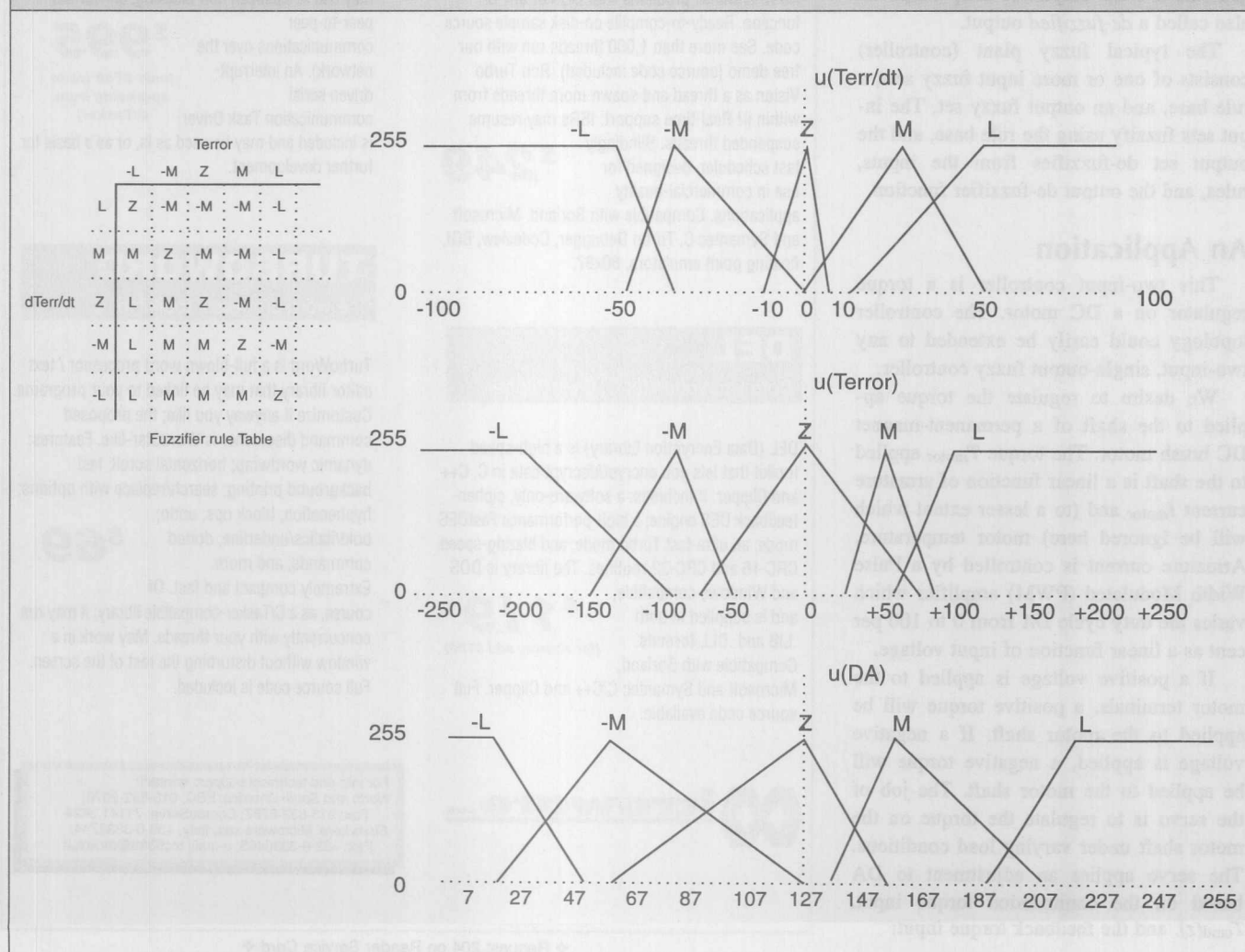
but the feedback torque derivative is non-zero:

$$T_{\text{arm}}(z) > T_{\text{arm}}(z-1)$$

we might wish to apply a slight adjustment to DA so that $T_{\text{error}}(z)$ and $dT_{\text{err}}(z)/dt$ remain zero. The amount of adjustment applied to DA is determined by the inputs and also the output control fuzzy set $\mu(DA)$. Which membership function of $\mu(DA)$ to apply is determined by the input membership function space and the operator-inferred description of the control problem in the rule-base table. The idea is to maintain zero error and zero error derivative:

$$dT_{\text{err}}(z)/dt = T_{\text{error}}(z) = 0$$

Figure 4 Fuzzifier rules and membership functions for 8-bit system



Your_Ideas += Our_Library();

Take your C or C++ code and add our library. The result? You get professional, high-performance applications complete with pick lists, dialog boxes, moving bar menus, data entry screens, automatic mouse support text and memo editor, and access to dBASE database files.

If you know dBASE, you know TOPAZ for C
TOPAZ's comprehensive library of high level, dBASE-like commands for Borland, Turbo, and Microsoft C/C++, lets you write code with calls to *Use()*, *Skip()*, *IndexOn()*, *Find()*, *Browse()*, *ReportForm()*

"An excellent product ... saved us thousands of hours of programming time..."

Robert Stanton
Hero Computers
Victoria, Australia

and over 530 other functions. For example, with TOPAZ you can open any dBASE database file and browse the data with just two simple lines of C. TOPAZ calls are always concise, error-free, easy to read, and simple to maintain.

Programmer Friendly

TOPAZ comes with a disk full of sample programs, a template-driven code generator, and handy

utilities for database definition, report generation, even color mapping, and 800 pages of documentation and examples. Run the menu-driven installation program, and you can be running your first TOPAZ program in minutes.

"Let's us accomplish our mission."

Wayne Spring
Hughes Aircraft
Las Vegas, NV

Our ironclad guarantee:
If for any reason you aren't completely satisfied with Topaz, return it to us for a prompt refund.



TOPAZ[®]

FOR C
\$299

Order now and while supplies last, we'll be happy to include in your order a copy of *SourcePrint*, a \$99 value, absolutely free.

	TOPAZ	CodeBase	Clipper	FoxPro
dBASE style syntax	✓	No	✓	✓
Over 530 functions	✓	No	No	✓
Full screen validated data entry	✓	No	✓	✓
Easy pick and tag lists	✓	No	No	No
Dialog boxes and progress bars	✓	No	No	No
Virtual fields and files	✓	✓	No	✓
Nested Browse sessions	✓	No	No	✓
Fast non-indexed search	✓	✓	No	No
Page image printing	✓	No	No	No
End-user help system	✓	No	✓	✓
Time math functions	✓	✓	No	No
Pop-up interactive calendar	✓	No	No	✓
Pop-up calculator	✓	No	No	No
Automatic mouse support	✓	No	No	✓
Report generator	✓	✓	✓	✓
Code generator	✓	No	No	✓
Create stand-alone EXE files	✓	✓	✓	EXTRA COST
Build multi-user programs	✓	✓	✓	✓
Database engine for Windows	✓	✓	No	✓
Source code included	✓	✓	No	No
Price	\$299	\$495	\$495	\$795

To order: Call toll-free: 800-468-9273 (orders only please). For information and international orders: 415-697-0411. All orders add \$6 U.S. shipping and handling; \$12 in AK, HI, and Canada; \$25 international. Calif. residents add 8 1/4% sales tax. Microsoft Windows™ support included. Dealers: TOPAZ is available from Software Resource.

S O F T W A R E S C I E N C E I N C .



The number of inputs and the dimension of each determines the rule base matrix. For example, we have two input fuzzy sets, $\mu(T_{Verr})$ and $\mu(dT_{err}/dt)$, with five membership functions each. This yields a 5x5 or 25 rule-base controller. The corresponding action to be performed by the evaluation of each input rule determines the output controlling membership function $\mu(DA)$. For example, rule (3, 3) is:

```
IF  $T_{error} == M$ 
  AND  $dT_{err}/dt == -M$ 
  THEN  $DA == Z$ 
```

and rule (3, 2) is:

```
IF  $T_{error} == M$ 
  AND  $dT_{err}/dt == Z$ 
  THEN  $DA == -M$ 
```

A literal evaluation of rule (3,2) would read, "If the feedback torque is less than the setpoint, and the feedback torque is approaching the setpoint at medium rate, then apply zero bias to the PWM." The key to fuzzy logic control is that the degrees of membership incorporate the amount of bias to apply and the degree of truth to each rule. (See Figure 4.)

QNX[®] 4.2 Operating System

Realtime 32 Bit POSIX Compliant OS

Multitasking, Multiuser, Peer-to-peer Integrated Networking within a 7K Microkernel

► Development Environment:

WatCom ANSI C Compiler, 32 Bit
WatCom C++

► Realtime Scheduler: 3 Options

► Network Support: Arcnet and Ethernet

► Database: Xbase, Btree, SQL and others

► Connectivity: TCP/IP and DOS-QNX links

► Graphics: Direct Graphics and GUI Options

► DOS emulation within QNX supporting Windows 3.1

► Configurable for Embedded Applications

©QNX is a registered Trademark of QNX Software Systems Ltd.

Call or write for more information:

T&T COMPUTER PRODUCTS, INC

P.O. Box 14010 Tulsa, OK 74159 USA
Tel: 918/742-1816 Fax: 918/749-7113
VISA/MC

◆ Request 247 on Reader Service Card ◆

When a Rule Fires

A rule *fires* when a nonzero result is returned upon evaluation of the rule. The degree to which the rule is true is incorporated in the membership functions. If the input values lie within the membership function trapezoid space, then the input is valid. A rule fires when both inputs to the rule are non-zero. A rule is evaluated using ternary operators similar in concept to Boolean operators. The operators logically follow the AND, OR, and NOT constructs.

Listing 1 C declaration code for rule table and membership functions

```
/******
   File: fuzzy.h
   Date: 4/3/93
   Author: Jack J. McCauley
   Header file for fuzzy.c
   *****/

/* membership function size */
#define TORQUE_MEMBERS 5
#define DER_MEMBERS 5

/* loop frequency */
#define LOOP_HZ 1

/* integrator constant z-1*/
#define SKIP 1

/* normalized value */
#define NORMAL 255

/* Current ma */
#define MAX_TORQUE 1000
#define MIN_TORQUE -1000

/* derivative */
#define MAX_DERI LOOP_HZ*MAX_TORQUE
#define MIN_DERI LOOP_HZ*MIN_TORQUE

/* pwm 1/10 % */
#define MAX_PWM 255
#define MIN_PWM 0

/* Max and Min */
#define MIN_ERROR 0
#define MAX_ERROR 255

/* size of all arrays */
#define ARRAY_SIZE 256

/* represent a normalized 0 -> 1000 = 0.0 -> 1.0 */
#define FUZZY_RADIX NORMAL

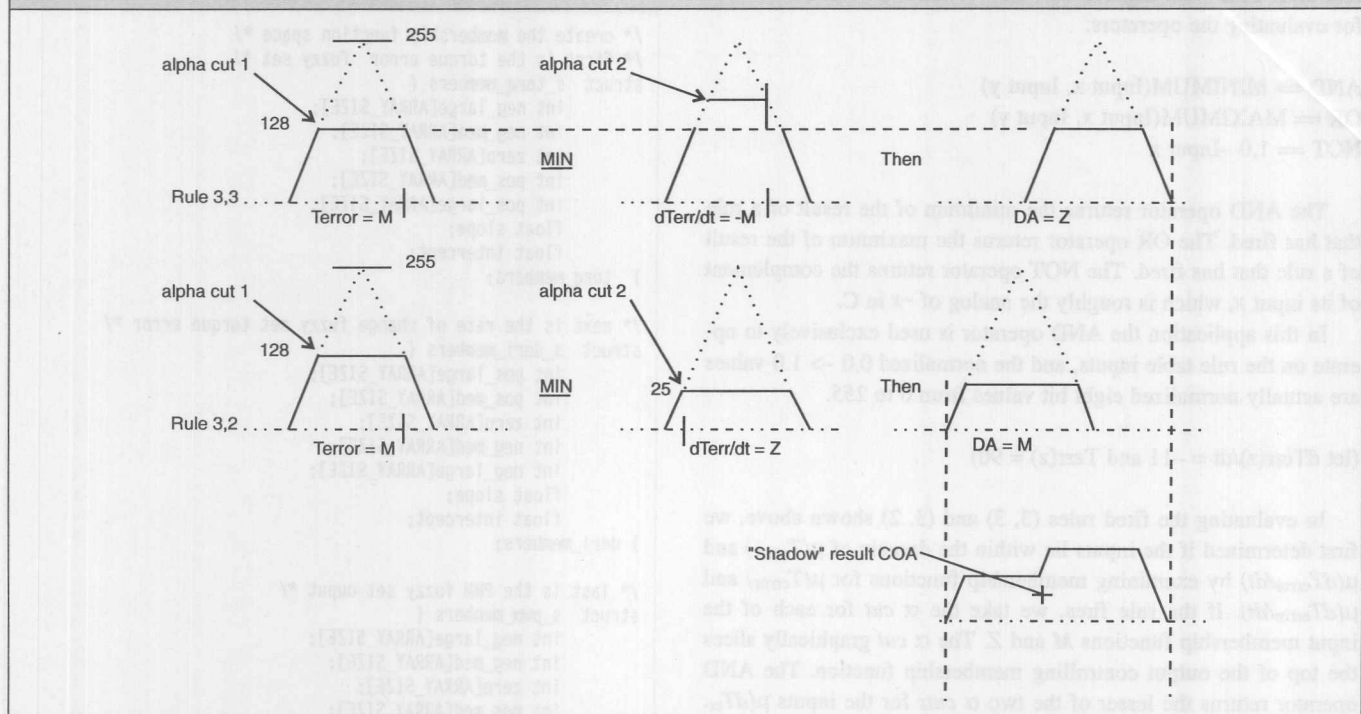
/* fuzzy max (ternary) operator */
#define FUZ_MAX( x, y ) ((x>y) ? x : y)

/* fuzzy min (ternary) operator */
#define FUZ_MIN( x, y ) ((x<y) ? x : y)

/* fuzzy AND operator */
#define FUZ_AND( x, y ) FUZ_MIN( x, y )

/* fuzzy OR operator */
#define FUZ_OR( x, y ) FUZ_MAX( x, y )

/* fuzzy compliment operator */
#define FUZ_NOT( x ) ( FUZZY_RADIX - x )
```

Figure 5 Example of Fuzzify \geq Defuzzify when two rules fire

Windows Communications

Do you need to make your Windows Application communicate over a COM port? Need to transfer data files from your Access, Paradox, FoxBase, or SQL Base Windows Data Base? Need to make your Visual Basic, Turbo Pascal, SmallTalk, Power Builder, or Object Vision application interface with a COM port? Now you can do all these things and more using CrystalCOMM through our dynamic linked library interfaces.

CrystalCOMM for Windows \$175

CrystalCOMM for NT \$200

CrystalCOMM is written in C for the Windows and NT environments. It supports the development of modem or serial port communication programs. It includes everything to maintain consistency in the Windows foreground or background environment and includes DLL support. CrystalCOMM supports XMODEM, XMODEM-CRC, XMODEM-1K, YMODEM, YMODEM/Batch, ZMODEM, KERMIT, and ASCII protocols through a simple, structured function interface. Supports both packet or file calls. Supports Digiboard and up to nine simultaneous ports at high speed. Detailed examples in C, Turbo Pascal, Visual Basic, and Access. Library includes object, source (optional), and documentation. Also available for DOS at \$125.



Crystal Software, Inc. (906)822-7992

P. O. Box 247, Amasa, MI 49903, USA - FAX (906) 822-7994

STOP THE PIRATES®! USE THE PROVEN PLUG

The proof is in our thousands of satisfied customers - You too deserve the best protection at the lowest price!

EliaShim's Software Protection Systems:

MEMOPLUG®

Protection based on a combination of a unique code, a serial number and a version number.

FILES OPTION®

Memoplug-based protection for NON-executable programs, such as AUTOCAD, LOTUS, interpreter data files, run time modules, macros, LISP.

LANPLUG®

Complete protection for a network with one plug. Limits number of users.

New!
Lease your software with
CLOCKPLUG®
Call for details and availability

EliaShim
MICROCOMPUTERS INC

Call Now: 1-800-677-1587

4005 Wedgemere Dr.
Tampa, FL 33610
TEL.: (813) 744-5177 FAX: (813) 744-5197

◆ Request 144 on Reader Service Card ◆

Rules and rule bases can be built using the operators exactly like those in standard Boolean logic, but because membership functions deal with degrees of truth we must have a mechanism for evaluating the operators:

AND == MINIMUM(Input x, Input y)

OR == MAXIMUM(Input x, Input y)

NOT == 1.0 -Input x

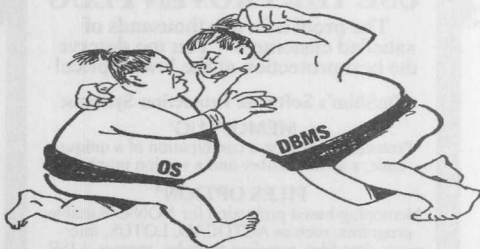
The AND operator returns the minimum of the result of a rule that has fired. The OR operator returns the maximum of the result of a rule that has fired. The NOT operator returns the complement of its input x , which is roughly the analog of $\sim x$ in C.

In this application the AND operator is used exclusively to operate on the rule table inputs, and the normalized 0.0 \rightarrow 1.0 values are actually normalized eight bit values from 0 to 255.

(let dTerror/dt = -11 and Terror(z) = 90)

In evaluating the fired rules (3, 3) and (3, 2) shown above, we first determined if the inputs lie within the domain of $\mu(T_{\text{error}})$ and $\mu(dT_{\text{error}}/dt)$ by examining membership functions for $\mu(T_{\text{error}})$ and $\mu(dT_{\text{error}}/dt)$. If the rule fires, we take the α cut for each of the input membership functions M and Z . The α cut graphically slices the top of the output controlling membership function. The AND operator returns the lesser of the two α cuts for the inputs $\mu(dT_{\text{error}}/dt, L)$ and $\mu(T_{\text{error}}, M)$ for use in the slicing operation. Because

REAL-TIME SUMO?



Big, fat data base management systems don't like sharing their space with an operating system. That's bad news for real-time developers, because conflict and duplication result in the loss of both speed and predictability.

Zip_RTDBMS is a POSIX-compatible real-time DBMS that works with a real-time operating system to deliver superior data base management performance and predictability.

Call DBx, Inc. today. Ask for the **Zip_RTDBMS** introduction packet. Royalty-free licensing is available.

Call today for a **FREE** copy of "Real-Time Data Management: the Nuts and Bolts of Predictable Information Computing".

DBx, Inc.

"The Advantage is Speed"

PO Box 8446
Cherry Hill, NJ
08002-0446 USA
(609) 667-9322 (+Fax)

© 1993 DBx, Inc.

Listing 1 continued

```
/* create the membership function space */
/* first is the torque error fuzzy set */
struct s_torq_members {
    int neg_large[ARRAY_SIZE];
    int neg_med[ARRAY_SIZE];
    int zero[ARRAY_SIZE];
    int pos_med[ARRAY_SIZE];
    int pos_large[ARRAY_SIZE];
    float slope;
    float intercept;
} torq_members;

/* next is the rate of change fuzzy set torque error */
struct s_deri_members {
    int pos_large[ARRAY_SIZE];
    int pos_med[ARRAY_SIZE];
    int zero[ARRAY_SIZE];
    int neg_med[ARRAY_SIZE];
    int neg_large[ARRAY_SIZE];
    float slope;
    float intercept;
} deri_members;

/* last is the PWM fuzzy set output */
struct s_pwm_members {
    int neg_large[ARRAY_SIZE];
    int neg_med[ARRAY_SIZE];
    int zero[ARRAY_SIZE];
    int pos_med[ARRAY_SIZE];
    int pos_large[ARRAY_SIZE];
    float slope;
    float intercept;
} pwm_members;

/* define output rule table members */
#define L      pwm_members.pos_large
#define M      pwm_members.pos_med
#define Z      pwm_members.zero
#define NM     pwm_members.neg_med
#define NL     pwm_members.neg_large

/* finally the fuzzy sets */
/* torque feedback error */
int *dTerror_dt[] = {
    deri_members.pos_large ,
    deri_members.pos_med ,
    deri_members.zero ,
    deri_members.neg_med ,
    deri_members.neg_large
};

/* torque (OZ-in) */
int *Terror[] = {
    torq_members.neg_large ,
    torq_members.neg_med ,
    torq_members.zero ,
    torq_members.pos_med ,
    torq_members.pos_large
};

/* create the rule table and allocate space */
struct s_rule {
    int *table[TORQUE_MEMBERS];
} rule[DER_MEMBERS] =
/* */
{{Z,  NM, NM, NM,  NL },
 {M,   Z,  NM, NM,  NL },
 {L,   M,  Z,  NM,  NL },
 {L,   M,  M,  Z,  NM },
 {L,   M,  M,  M,  Z }};
```




SERIAL I/O

New Version

COMM-DRV™

New Version

COMM-DRV is a professional serial communication library for **Windows** and **MS-DOS**. In addition to the libraries, it comes with an extensive set of utilities, including a true Windows replacement driver, a true MS-DOS device driver, several TSRs, DLLs, debugging monitor, and much more. **COMM-DRV** may be used with any Language, any Tool, any Database Language/Environment, and any Application. Do not be fooled by false claims. **COMM-DRV** will be the only serial communication development tool you need. When you purchase **COMM-DRV**, you inherit a team of experienced professional software engineers to help in your design. And you get a 30 day money back guarantee.

"Serial communication is at the core of what we do, and COMM-DRV has helped us rapidly develop new applications, and easily add multiple port features. We've been very impressed with the flexibility of COMM-DRV, and the technical support has been outstanding."

Lee Perryman, Deputy Director and head of technology development, Associated Press Broadcast Services, Washington, DC

- Transparent support for several smart cards(Arnet Smartplus series, Digiboard COMXi, PCXi, PCXm, PCXe).
- Support for all dumb multiport boards on the market.
- Asynchronous callback to user functions on different serial communication events(modem signals, buffer count, character reception and much more).
- Timed asynchronous callbacks to user functions.
- X, Y, & Zmodem on multiple ports concurrently.
- Complete source code to all libraries.
- Unlimited number of ports active concurrently.
- Real time serial port monitoring to screen and disk.
- Quickbasic, Visual Basic, C, Visual C++, Assembly, Microsoft Access examples.
- Hardware/Software Flow Control.
- Interrupt 14H/FOSSIL/EBIOS replacement.
- One API to learn (Same for Windows & MS-DOS).
- 100% Port re-entrant code(Important for multitasking).
- Desqview®, Procomm® Windows, PCBoard® BBS PcAnywhere® compatible.
- Integrate with any database tool, language, or application without writing any serial communication code.
- ANSI/BIOS screen management utility.
- TSR to redirect serial data to keyboard buffer, transparent to user application(Great for barcode & POS apps).
- High level Hayes compatible modem support.
- True Windows comm device replacement. Use your favorite Windows communication applications on all multiport cards we support.
- True DOS device driver that allows serial ports to be opened as normal files under MS-DOS or Windows.
- Transmit data from user callback on any event(Important for multidrop applications).
- 8250/16450/16550 Auto detection/Up to 115200 baud.
- Remap baud rates/Change baud rate divisor.
- Product customization.

COMM-LOG™

Group of cooperating programs and TSRs for developing serial applications quickly and efficiently. Programs and TSRs may be called directly from applications by shelling to them or by using a "C" API. Alternatively, applications may be built entirely with English like scripts.

COMM-LOG Features:

- TSR that log serial data to disk entirely in the background concurrently on any number of ports. Data may be extracted from the file while logging is in progress.
- TSR to perform X/Y/Zmodem file transfers entirely in the background concurrently on any number of ports.
- Full featured script engine that can function standalone or that may be called from a user application. Scripts may be run in the background.
- Extensive set of examples.

MTASK™

Multitasking kernel used in the development of multi-threaded applications and TSRs that run in the background. Its used commercially to create background TSRs for fax, communication, data acquisition, and many other applications. All tasks may call DOS functions.

MTASK Features:

- Create and destroy tasks dynamically.
- Put tasks to sleep for specific duration.
- Send/receive messages between separate threads in application and/or between TSRs running in the background.
- Re-entrant libraries that include functions that make creating TSRs as easy as a C or Assembly function call.
- Extensive set of example that show the ease of writing TSR that run entirely in the background.

COMM-DRV	\$189.95
COMM-LOG.....	\$189.95
MTASK	\$299.95
COMM-DRV, MTASK Combo	\$399.95
COMM-DRV, MTASK, COMM-LOG Combo ...	\$499.95

4 Port Multiport Card(16450)	\$110.00
4 Port Multiport Card(16550A)	\$170.00
8 Port Multiport Card(16550A)	\$225.00

Company: WCSC
Address: 2470 S. Dairy Ashford Suite 188,
Houston, TX, 77099

Telephones: 1-800-966-4832
713-498-4832

Fax 713-568-3334
BBS 713-568-6401

Visa/Mastercard/American Express/Optima/Discover
P.O./Checks/Wire transfers

Listing 1 *continued*

The Fuzzy membership functions are stored in a look up table for all input and output fuzzy sets. Fuzzy variables with three digits of precision are normalized between 0 and 255 integer corresponding to an 8-bit radix converter and a fuzzy set 0.0 - 1.0 normalized value. Look up tables increase bandwidth and allow the servo to run with few floating point calculations.

```

/*****
File: fuzzy.c
Date: 4/3/93
Author: Jack J. McCauley
fuzzy torque controller for motor
*****/

#include "stdio.h"
#include "stdlib.h"
#include "conio.h"
#include "proto.h"
#include "fuzz.h" /* the above file */

/*****
Routine: calc_slope
Date: 4/3/93
Author: Jack J. McCauley
calcs line slope of two points in a plane
*****/
float calc_slope( float x1, float x2, float y1, float y2 ) {

    float slope;

```

of the overlapping domain of the membership functions, more than one rule will typically fire. When this occurs, fuzzy logic provides a mathematical method for dealing with multiple rule firings. The output of each rule is an output membership function that has been α cut and is then graphically ORED with the accumulated preceding output membership functions. The resultant resembles a "shadow" of each fired rule with the accumulated output overlapping each preceding fired rule. Mathematically, the accumulation is the fuzzy OR operator (MAXIMUM) applied to the current rule that fires and the accumulated fired rules. (See Figure 5.)

De-fuzzification

The accumulated result of all firing rules occupies a two-dimensional space which is a fuzzy set with one membership function. The resultant must be converted to a real world crisp result which takes into account all applied rules. To do this, a spatial average is taken which computes the center of area (COA) of the function. COA is among several methods for computing the average. It is the method used in this application because it executes quickly.

$$COA = (25*18 + 25*31 + 128*100 + 128*127 + 128*140) / (25 + 25 + 128 + 128 + 128) = 74$$

So the DA value written to the PWM D/A is 74.

C Language Implementation

The membership functions use table look up to follow the example:

```

μ(dTerr/dt) (in oz-in):
-L WHEN dTerr(z)/dt <= -35
-M WHEN -50 <= dTerr(z)/dt <= 0
-ZM WHEN -10 >= dTerr(z)/dt <= 7
-M WHEN 0 <= dTerr(z)/dt <= 50
-L WHEN dTerr(z)/dt >= 10

```

All 25 rules are evaluated and the rules that fire result in a output according to the COA output de-fuzzifier function. The output array stores the result of the rule firing and the history of predicate rules. Conflicts are resolved using the fuzzy OR (MAXIMUM) operator. The resultant outputs contribute to a weighted sum to yield the COA output. This strategy allows for the quick computation of a crisp value using eight-bit multiplies.

Real-Time Multitasking with DOS

for Microsoft C, Borland C, Borland/Turbo Pascal

Develop Real-Time Multitasking Applications under MS-DOS with RTKernel!

RTKernel is a professional, high-performance real-time multitasking kernel. It runs under MS-DOS or in ROM and supports Microsoft C, Borland C++, Borland/Turbo Pascal, and Stony Brook Pascal*. RTKernel is a library you can link to your application. It lets you run several C functions or Pascal procedures as parallel tasks. RTKernel offers the following advanced features:

- pre-emptive, event/interrupt-driven scheduling
- number of tasks only limited by available RAM
- task-switch time of approx. 6 μ secs (33-MHz 486)
- performance is independent of the number of tasks
- use up to 64 priorities to control your tasks
- priorities changeable at run-time
- time-slicing can be activated
- programmable timer interrupt rate (0.1 to 55 ms)
- high-resolution timer for time measurement (1 μ sec)
- activate or suspend tasks out of interrupt handlers
- programmable interrupt priorities
- supports math coprocessor and emulator
- semaphores, mailboxes, and message-passing
- keyboard, hard disk, and floppy disk idle times usable by other tasks
- interrupt handlers for keyboard, COM ports, and network interrupts included with source code
- supports up to 36 COM ports (DigiBoard, Hostess boards)
- full support of NS16550 UART chip
- fast, inter-network communication using Novell's IPX
- runs under MS-DOS 3.0 to 6.0, DR-DOS, LANs, or without operating system
- DOS calls from several tasks without re-entrance problems
- supports resident multi-tasking applications (TSRs)
- runs Windows or DOS Extenders as a task
- supports CodeView and Turbo Debugger
- ROMable
- full source code available
- no run-time royalties
- free technical support by phone or fax

RTKernel-C (MSC 6.0/7.0/8.0, BC++ 1.0/2.0/3.x) \$495 (Source Code: add \$445)

RTKernel-Pascal (TP/BP 5.x/6.0/7.0, SBP 6.x) \$445 (Source Code: add \$375)

For international orders, add \$30 for shipping and handling. MasterCard, Visa, check, bank transfer, or COD accepted.

FREE DEMO DISK
Please request Info Kit C

On Time
MARKETING

Professional Programming Tools

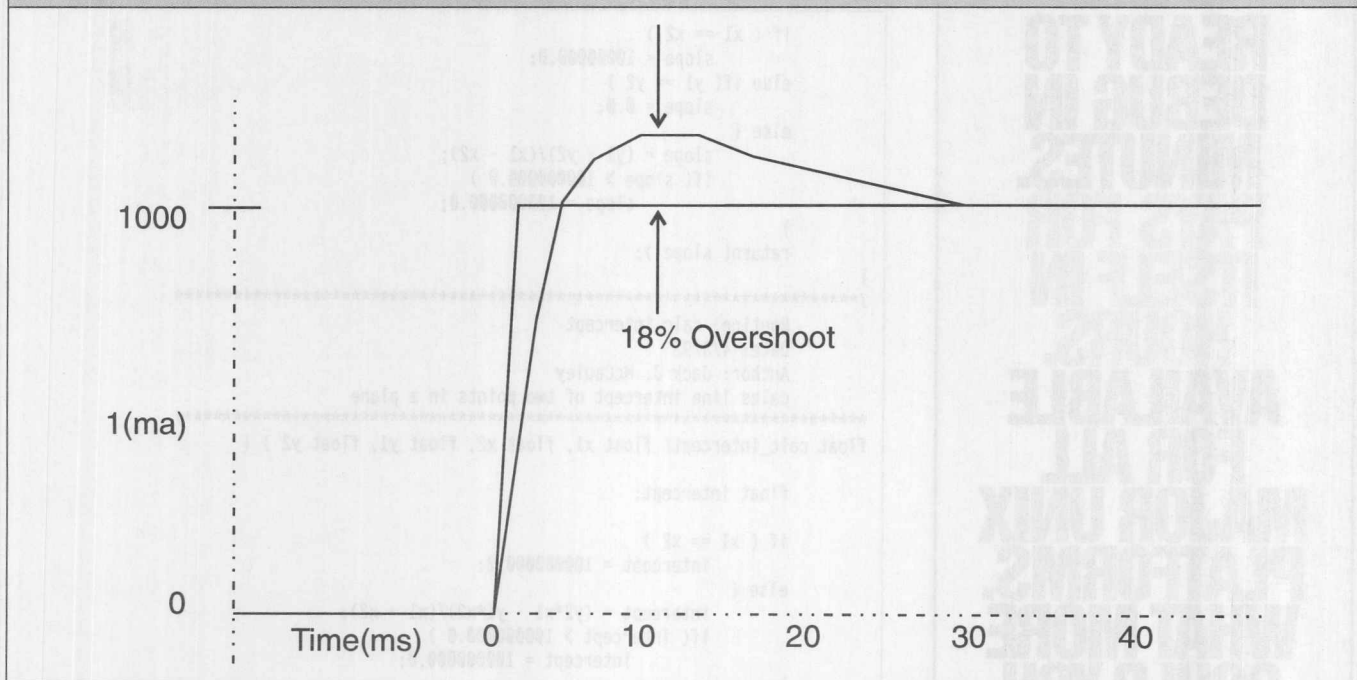
In North America, please contact:

LEL Computer Systems
20 Canterbury Court
Selauket, NY 11733 • USA
Phone (516) 473-8119 • Fax (516) 331-0706

Outside North America, please contact:

On Time Marketing
Karolinenstrasse 32 • 20357 Hamburg • GERMANY
Phone +49 - 40 - 43 74 72 • Fax +49 - 40 - 43 51 96
CompuServe 100140.633

◆ Request 214 on Reader Service Card ◆

Figure 6 An illustration of de-fuzzification

CODE RUNNER

**"Makes your C TSRs
smaller than you
could imagine."**

Mark Davidson, Computer Language

CodeRunnerR The #1 TSR Library for C and ASM

- 300+ functions in highly optimized assembler
- Auto-Disposal of initialization code and data
- Hotkeys, schedulers, fast background COMs
- Safe DOS use from TSRs, Network Friendly
- Online hypertext help, Quick start templates
- Swap apps or graphics to EMS/XMS, spawn ...

Pick of the Professionals

"cream of the crop"
Tom Swan, PC World

"highly useful"
R. Bradley Andrews, DDJ

"not only solves problems, it inspires new possibilities.
It is destined for the programmer's Hall of Fame"

Joe Campbell, Author C Programmer's Guide to Serial Communications

"the size of any program you create with CodeRunnerR
will astound you"

Mark Davidson, Computer Language

"excels in its TSR capabilities, coexistence with
other DOS applications and support"

Victor R. Volkman, The C Users Journal

"professional development tool that'll let you create
compact, fast TSRs"

Gary Entsminger, Micro Cornucopia

Point
Omega Point, Inc.

Omega Point, Inc.
25 Birch Rd., Framingham, MA 01701
508 877-1819 FAX: 508 877-0915

**Locked into old
PASCAL or BASIC?**
Wish Your Software Was In
C ?

Then Don't Re-Invent the Wheel !

**Automatically translate your code into readable
and maintainable C with
PASCAL and BASIC to C Translators**

*Available for most popular variants
eg. Turbo Pascal, VAX Pascal/Basic, Microsoft Pascal/Basic*

For more information call now!

Technosoft (US)
PO Box 8210
Rockford, IL 61126-8210
Phone: 815-397-3214

Technosoft (Europe)
Clarendon Court
Stockbridge, SO20 8HU. UK
Phone: +44-264-781626



All Registered Trade Marks Acknowledged



◆ Request 136 on Reader Service Card ◆

◆ Request 250 on Reader Service Card ◆

**READY TO
DEBUG IN
MINUTES.
PAYS FOR
ITSELF IN
HOURS.
AVAILABLE
FOR ALL
MAJOR UNIX
PLATFORMS.
WHAT MORE
COULD YOU
ASK?
OKAY. TRY IT
FOR 15 DAYS
FREE.**

SENTINEL: The Single Choice For Multi-Platform Memory Debugging.

There's only one way to develop quality software for heterogeneous environments. Debug with SENTINEL.

SENTINEL saves costly program repairs by detecting memory errors, leaks and run-time bugs *before* you ship your product.

In fact, developers for HP, Sun, SGI, IBM, DEC, DG, and other popular UNIX platforms find that SENTINEL can pay for itself the first time you use it. See for yourself. Call today for a FREE 15-day trial. You have nothing to lose. Valuable time to save. And better software to gain.

What more could you ask?

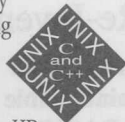
1-800-296-3000

AIB SOFTWARE CORPORATION

Formerly Virtual Technologies, Inc.

46030 Manekin Plaza, Suite 160, Dulles, VA 20166
(703) 430-9247, E-mail: info@vti.com, FAX(703) 450-4560

©Copyright 1993, AIB Software Corporation
Registered trademarks are proprietary to their respective manufacturers.



Listing 1 *continued*

```

if ( x1 == x2 )
    slope = 10000000.0;
else if( y1 == y2 )
    slope = 0.0;
else {
    slope = (y1 - y2)/(x1 - x2);
    if( slope > 10000000.0 )
        slope = 10000000.0;
}
return( slope );
}
/*****
Routine: calc_intercept
Date: 4/3/93
Author: Jack J. McCauley
calcs line intercept of two points in a plane
*****/
float calc_intercept( float x1, float x2, float y1, float y2 ) {

    float intercept;

    if ( x1 == x2 )
        intercept = 10000000.0;
    else {
        intercept = (y2*x1 - y1*x2)/(x1 - x2);
        if( intercept > 10000000.0 )
            intercept = 10000000.0;
    }
    return( intercept );
}
/* end print_array */

/*****
Routine: down_load()
Date: 4/3/93
Author: Jack J. McCauley
initialize fuzzy membership function tables from serial port DRIVER not shown
*****/
void down_load( int *membership_function, int len )
{
    int k;
    char buff[32];

    /*short, tight loop */
    for( k=0; k< len; k++ ) {
        /* get ascii string from driver */
        gets( buff, 12 );
        *membership_function++ = atoi( buff );
    }
}
/* end down_load() */
/*****
Routine: fuzzy_init
Date: 4/3/93
Author: Jack J. McCauley
initialize fuzzy membership function tables
*****/
void fuzzy_init( void )
{
    int k;
    long slope;
    long intercept;
    int val;

    /*
There are several ways to generate these tables:

1) The easiest is to simply use ROM space and store them permanently in memory. FUZZ.H
would need to be modified to reflect the static ROM delecations for each fuzzy
set and the values would be initialize directly attaching them to the data arrays.

```

◆ Request 150 on Reader Service Card ◆

For a micro-controller application, It was determined by experimentation that a loop repetition rate of at least 1 KHz would be required for satisfactory performance. Command set-points were entered through a serial port and an oscilloscope measured the response of the loop. Figure 6 shows the measured step response, which is quite adequate.

Conclusion

Fuzzy logic control seems to hold promise as an alternative to standard PID control. Rigid modeling is not required because the servo tuning is done via the fuzzy set membership functions and the rules. The rules incorporate Boolean statements and operator-inferred control rather than strict theoretical modeling. The membership functions allow tuning of the control system by altering the space that they occupy, and the placements of the output membership functions provide the control action to be performed by the rule. The rules use Boolean-like operators that infer a control action depending upon the degree of truth of the rule firing. A rule fires if the fuzzified inputs return a non-zero degree of truth, referred to as the *a cut* for each input.

Fuzzy logic is not as brittle as standard PID control because of the natural language approach to the control problem. Second-order effects such as temperature can be incorporated directly in the rule by logically appending another fuzzy set for temperature onto the rule. Suppose that we know that an increase in temperature causes the torque to change much more rapidly. By creating a rule base that incorporates temperature, we compensate for the change, as in:

```
IF  $T_{error} == M$ 
  AND  $dT_{err}/dt == -M$ 
  AND  $Temperature == L$ 
  THEN  $DA = M$ 
```

In PID control second-order variables such as temperatures are not so easily incorporated into the model.

The fuzzy controller is also less prone to radix overflow and underflow problems. In this application, the controller was implemented using eight-bit LUTs. All mathematical operations were cast to 32-bit *long*, thus preventing overflow. Because of the averaging nature of COA, it is inherently safe from overflow. □

FINALLY ! Hi-Res Graphics and Formatted Text for Screen and Any Printer

PRINT TOOLS LIBRARY C/C++ or BASIC (Inc. source code).....\$350
AT LAST! Make your application device independent . . . without having to spend weeks or months in development time. One set of code supports text and graphics (B/W and color) at full resolution for screen, laser, dot matrix, DeskJet, PaintJet, etc. Print Tools adds windows like functionality to your DOS programs. Great for graphs, pie charts, and forms. Set type justified, centered, flush right. Even a dot matrix achieves laser printer results.

PCL TOOLKIT C/C++ or BASIC (Inc. source code).....\$215
If your application supports PCL printers only, then the PCL ToolKit provides you with an easy and fast way to incorporate the PCL escape sequences into your program. Functions include: setting type centered, justified, flush right for both bitmapped and internal scalable fonts. Draw lines, circles, ellipses, arcs, pies, and 3-D bar shapes and fills. Add logos and pictures or PCX graphics.

DESIGN-A-FORM III™ with print preview.....\$215
Integrate complex forms into your program without having to learn a single line of PCL. Stand alone, menu driven program . . . easy to use. Include logos/graphics, repeat check-off boxes across and down with a single instruction, copy and merge all or part of a form, access scalable fonts of the HP III/4, import ASCII files and reduce the vertical spacing of the entire form as a single instruction. Plus get 6 fonts FREE. There is no royalty fee for distributing your forms or fonts. You are in total control.

SCALABLE FONTS

FONT MAKER LIBRARY C/C++ or BASIC.....\$290
This is not just a library for creating soft fonts, but a tool to create fonts from within your program. Download directly 'on the fly' or use the EXE program. Distribute fonts with your program — **royalty free**. Create any point size, bold and italic. Also, use with DESIGN-A-FORM and Print Tools. Includes one Font Pak, Roman or Helvetica. Additional Standard Font Pak \$75. Custom Font Paks available.

FONT CONSTRUCTION SET.....\$490
Our state of the art technology vectorizes a bitmapped font into a scalable font. Transform any of the hundreds of bitmapped fonts into your own scalable font. No royalty fees for distribution. If your applications require many typefaces and you want control over typeface creation, then the Font Construction Set is what you need.

INSTALLATION SOFTWARE

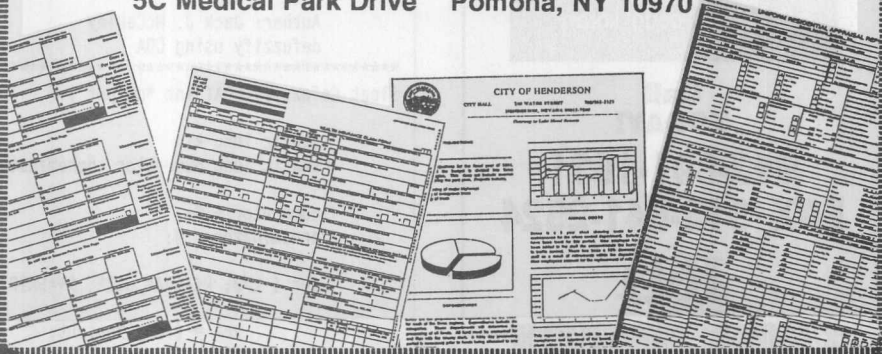
ProStall Ver. 3.0.....\$169
ProStall Plus (with copy protection).....\$269
First impressions count. So, if you distribute software to end users, give your product that professional image. Compress files and copy large files across multiple disks. Your company name appears during installation. Supports unlimited tree structure. You limit the number of installs if desired. Include up to two text files. Add PATH, FILE and BUFFER statements. Menu driven, easy to use — save time and money.

BYTECH

(914) 354-8666

BUSINESS SYSTEMS, INC.

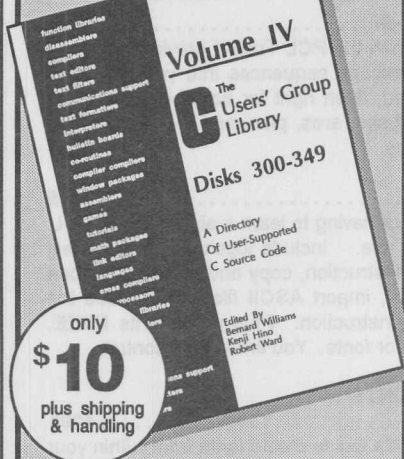
5C Medical Park Drive Pomona, NY 10970



◆ Request 229 on Reader Service Card ◆

- cross compilers
- function libraries
- tutorials
- games
- disassemblers
- communications
- compilers
- languages
- and more

**NOW
AVAILABLE**



**CUG Library
Directory
Volume IV
(Volumes 300-349)**

NEW

- volume cross reference by topic

EXPANDED

- keyword index
- reviews of key volumes
- capsule summaries of ALL volumes

The complete reference to volumes 300-349 of the CUG public domain and shareware C source code library

**Call
TODAY!
913-841-1631
FAX 913-841-2624**

R&D
publications, inc.

♦ Request 277 on Reader Service Card ♦

Listing 1 *continued*

2) Down load the fuzzy sets through the serial port as I did in the tuning of this servo. In which case I've included that code here (of course in the ROM version of the system you'll need it)).

3) Store the slope-intercept form of each membership function and calculate the line slopes and intercepts

*/

/*

Membership functions are downloaded through the serial port from a file in ASCII format one set member at a time. In this system I used a spreadsheet and graphical interface to actually draw the membership functions with a mouse. This aided in tuning the servo substantially. In the ROM version of the system, I wrote a small program to append the ROM statics memberships to fuzz.h.

*/

```
download( deri_members.neg_large, ARRAY_SIZE);
download( torq_members.neg_large, ARRAY_SIZE);
download( pwm_members.neg_large, ARRAY_SIZE);
download( deri_members.neg_med, ARRAY_SIZE);
download( torq_members.neg_med, ARRAY_SIZE);
download( deri_members.zero, ARRAY_SIZE);
download( torq_members.zero, ARRAY_SIZE);
download( pwm_members.zero, ARRAY_SIZE);
download( deri_members.pos_med, ARRAY_SIZE);
download( orq_members.pos_med, ARRAY_SIZE);
download( pwm_members.pos_med, ARRAY_SIZE);
download( deri_members.pos_large, ARRAY_SIZE);
download( torq_members.pos_large, ARRAY_SIZE);
download( pwm_members.pos_large, ARRAY_SIZE);
```

/* slope intercept line calculation */

/* these line equations are used for scaling the
crisp values from the above arrays */

```
deri_members.slope = calc_slope(MIN_DERI, MAX_DERI, 0, ARRAY_SIZE-1);
deri_members.intercept = calc_intercept(MIN_DERI, MAX_DERI, 0, ARRAY_SIZE-1);
```

```
torq_members.slope = calc_slope(MIN_TORQUE, MAX_TORQUE, 0, ARRAY_SIZE-1);
torq_members.intercept = calc_intercept(MIN_TORQUE, MAX_TORQUE, 0, ARRAY_SIZE-1);
```

```
pwm_members.slope = calc_slope(0, ARRAY_SIZE-1, MIN_PWM, MAX_PWM);
pwm_members.intercept = calc_intercept(0, ARRAY_SIZE-1, MIN_PWM, MAX_PWM);
```

/*

```
for( k=0; k<ARRAY_SIZE; k++)
    printf("%d %d %d %d %d\n",k,deri_members.pos_large[k],deri_members.pos_med[k],
    deri_members.zero[k],deri_members.neg_med[k],deri_members.neg_large[k]);
*/
```

*/

}

/* end FUZZINIT */

Routine: defuzzify_COA

Date: 4/3/93

Author: Jack J. McCauley

defuzzify using COA

*****/

```
float defuzzify_COA( int *output ) {
```

```
static long k;
static long numerator, denominator, val;
```

```
numerator = 0;
denominator = 0;
```

```
for ( k=0; k<ARRAY_SIZE; k+=SKIP ) {
```

```
    val = (long)*output;
```




C MAGNA•COMM

VERSION 2.0

THE WORLD'S MOST COMPLETE C COMMUNICATIONS TOOLKIT.

Programmers call Magna•Comm C "the complete C communications toolkit" because it supports nearly every product and program with hundreds of powerful communications features. Magna•Comm C is the fast and easy way to add complete communications to all your applications.

Complete... FAX support - meets all CAS standards.

Complete... Serial communications.

Complete... Support for DOS and MS-Windows in a single package (Magna•Comm C for Windows). Use the same function calls in both environments.

Complete... With a terminal program application (similar to MS-Windows terminal) to show how to use Magna•Comm C under Windows or to add to or modify in your applications.

Complete... Support for nearly every program or product including MSC 5.1+/Quick C, Borland Turbo 3.0+, Watcom C and more.

Complete... Support for Z-modem and CompuServe QuickB File Transfer Protocol Suite.

Complete... Network modem support.

Complete... Diagnostics to determine if devices can share interrupts.

Plus, complete modem support, chip support, serial port support, flow control, terminal emulations, speeds up to 115,000 bps, and an extensive 600 page manual with 125 pages of background and 35 example programs.

ONLY \$229 (US) MAGNA•COMM C FOR DOS (PLUS SHIPPING)

Magna•Comm C for Extended DOS - \$299 (US)

Magna•Comm C for MS-Windows - \$299 (US)

Magna•Comm C Professional Combo - \$499 (US)

COMPLETE...
SOURCE CODE INCLUDED.
PLUS FULL-TIME TECH
SUPPORT.

For ordering and information call toll free

1-800-755-7344 (U.S. and
Canada)

Major credit cards accepted

SofDesign
INTERNATIONAL, INC.

C CODE FOR THE PC

source code, of course

	GraphiC 7.0 (high-resolution, scientific plots in color & hardcopy, contour plots, device independence)	\$370
	X/DOS and Xt/DOS (Xlib with X client and Xt toolkit for DOS; port X code to DOS; Xt/DOS requires X/DOS and 32-bit compiler)	each \$300
	ZIP Image Processor & Victor Image Library Version 2.2 (brightness, contrast, merge images, TIFF/GIF/PCX/bin, HP ScanJet support)	\$290
	The Snooper (Ethernet protocol analyzer for Novell NetWare and LAN Manager Networks; capture packets; real-time display)	\$275
	Embedded BIOS (full-featured, real-time input/output system; PC, XT, AT; for example, 80186 with 8259 interrupt controller)	\$260
	C/C++ Libraries by Code Farms (persistent C structures, ER models, dynamic arrays, database functions, Jolt Award winner; specify C or C++)	\$250
	TurboTeX (Release 3.0; HP, PS, dot drivers; CM fonts; LaTeX; MetaFont)	\$250
	Rogue Wave tools.h++ or math.h++ Class Library (extensive docs)	each \$240
NEW!	Crusher! (platform-independent data compression for network transfer; beats PK & LH on binary; directory trees; portable C)	\$215
	COMM-DRV (complete interrupt-driven serial communication libraries & device drivers; full source)	\$155
	TE Editor Developer's Kit Ver. 3.0 (full screen editor, undo command, multiple windows; with Word Processing \$230)	\$155
	Minix Operating System (Version 1.5; Unix-like operating system, includes manual; specify 5.25" or 3.5" diskettes)	\$150
NEW!	XASM (cross assemblers & utility programs; 65xx, 68xx, 80xx; Intel or Motorola hex format; macro preprocessor)	\$150
	Delorie GCC for MS-DOS (Version 2.2.2; includes C++, assembler, DOS extender, 387 emulation; complete source code and makefiles)	\$150
Updated!	Moby Crypto (PGP, DES, Secure Hash, UFC, MDs, Crack 4.1, Lucifer, IDEA, VCR+, large integer packs, tutorials, more; not for export)	\$150
	Lisp for DOS (Kyoto Common Lisp and CLISP; KCL includes Lisp-to-C translator for building mixed Lisp/C programs)	\$140
	Ibrow (Version 4.1; programmer's Windows-based editor; large files, help, undo/redo, drag-n-drop, function & type tags)	\$135
	Booster Toolkit (floppy disk bootstrap routines, DOS file system, light-weight multitasking, windows, fast memory management)	\$120
Updated!	SCM (portable Scheme in C, IEEE standard, includes JACAL symbolic math package; SCM-4D0/SLIB-1D5/JACAL-1A3)	\$100
	PC/IP (CMU/MIT TCP/IP for PCs; Crynwr drivers, NFS server, Bdale mailer, PCRoute/PCBridge, NDIS/ODI drivers, Beholder, more)	\$100
	DA (disassembler for Microsoft's New Executable (NE) binary files including Windows .exe, .drv, .dll, and .flt)	\$95
	Script Interpreter (a command script interpreter for DOS-based systems; C-like script language; lots of features)	\$90
	CELP 3.2c (Federal Standard 1016 Code Excited Linear Predictive voice sampling and encoding; voice over 4.8kbps; Unix code)	\$80
	CPPCOMM (C++ standard communications class library for DOS, Windows, OS/2, and NT; includes X/Y/Zmodem)	\$75
	ET Neural Net (back error propagation and Kohonen; specify DOS text, DOS VGS, or Windows)	\$75
	FlexList (doubly-linked lists of arbitrary data with multiple access methods; specify C or C++)	\$65
	Kier DateLib (all kinds of date manipulation; translation, validation, formatting, & arithmetic)	\$60
	Coder's Prolog (Version 3.0; inference engine for use with C programs)	\$60
Updated!	PCCTS Version 1.10 (Purdue Compiler Construction Tool Set; like YACC and LEX together with lots of additional features)	\$60
	Container Lite V 1.82 (C++ & FLC wrapper emulators; portable, persistent containers of arbitrary data including pointers)	\$50
	BigFloat (arbitrary precision floating point arithmetic and functions; includes BCD conversion)	\$50
	EZCalc (ASCII algebraic expression evaluator, unlimited parenthesis nesting, symbols, 32 built-in functions, easily extended)	\$50
	Backup & Restore Utility by Blake McBride (multiple volumes, file compression & encryption)	\$50
	CLIPS Version 6.0 (rule-based expert system generator; Windows compatible; hardcopy manuals additional)	\$50
	SuperGrep (exceptionally fast, revolutionary text searching algorithm; also searches sub-directories)	\$50
	OBJASM (convert .obj files to .asm files; output is MASM compatible)	\$50
	Editor Pack (20 public domain editors; micromacs 3.12, Stevie, Elvis, Moke, mg2a, DTE, Jove, Origami, CE & GRIEF)	\$50
	Exceptions for C (Ada-like exception handling for C programs; exceptions for any block; exceptions can be re-raised)	\$45
	DES Encryption & Decryption (2500 bits/second on 4.77 MHz PC for on-the-fly encryption at 2400 baud; not for export)	\$40
	Database Pack (9 databases - simple to complex: isam, bplus, AVL, SDB, ID, gdbm, Requiem, Ingres89, Postgres)	\$35
	COP (poor man's C++; C macro package which implements C++ in C)	\$35
NEW!	OCT (Object C Translator; essentially Brad Cox's Objective-C Version 4)	\$35
	RXC & EGREP Version 2.0 (Regular Expression Compiler and Pattern Matching; finite state machine from regular expression)	\$35
	Bison & BYACC (YACC workalike parser generators; documentation; includes C and C++ grammars)	\$35
	Spell Pack (6 spelling programs, a hyphenator, 2 utility packs and a 60K word list: Ispell, Microsp, Sp, Cspella, Spell, Dawg, Soundex)	\$30
	REGX Plus (Version 3.0, search and replace string manipulation routines based on compiled regular expressions)	\$30
	GNU Awk & Diff for PC (both programs in one package)	\$30
	Big Number Pack (7 arbitrary precision arithmetic packages in C, one in Fortran but free Fortran-to-C converter is included)	\$30
	CRUNCH Pack (30 file compression & expansion programs; now includes portable ZIP)	\$30
NEW!	OORT (C++ ray tracing code from the book by Nicholas Wilt)	\$30
	OEmacs (full GNU Emacs for DOS and Windows DOS box; C++ support, etags++, lots of .el files)	\$25
NEW!	CTask Version 2.22d (robust MS-DOS multitasking kernel; C functions run as light-weight processes; mailboxes, interrupts, pipes, etc.)	\$25
	PERL for MS-DOS (Version 4.019; C, sed, awk, and shell all rolled into one language; includes hardcopy docs)	\$25
Updated!	FLEX Version 2.4.3 (fast lexical analyzer generator; new, improved LEX)	\$25
	GNU RCS (FSF's version of the Revision Control System; like Unix's SCCS only better; keeps track of software development)	\$20
	Data	
	Moby Thesaurus II (6,000 root words, 2.5M synonyms, "common sense", concept related searches)	\$500
	Moby Pronunciator II (175,000 words & phrases encoded with full IPA pronunciation & emphasis points)	\$265
	Moby Part-of-Speech II (230,000 words and phrases described by prioritized part(s)-of-speech)	\$170
	Moby Hyphenator II (185,000 words fully hyphenated/syllabified)	\$105
	Moby Words II (610,000 words & phrases with Scrabble(tm) word list, place names, baby names, acronyms, core list for spell checkers)	\$100
	U. S. Cities (names & longitude/latitude of 32,000 U.S. cities and 6,000 state boundary points)	\$35
	The World Digitized (100,000 longitude/latitude of world country boundaries)	\$30
	CD-ROMs	
	BSD/386 (POSIX-compatible O/S; complete development package, full networking, kernel debugger, X11R5, DOS box; complete source code)	\$900
NEW!	AI CD-ROM (expert systems, neural networks, genetic algorithms, fuzzy logic, linguistics/natural language)	\$105
	FontMaster II CD Library (soft fonts for HP and HP compatible laser printers, 36 different type faces; 5,200 bit mapped fonts; 300MB)	\$70
Updated!	Prime Time for Unix (Volume 3, No. 1, January, 1994; over 6GB of Unix C code)	\$60
	Walnut Creek Libris Britannia (over 600MB of the best of British boards; not all source included)	\$55
NEW!	Mailer's Lookup (9-digit ZIP codes by street address or company name, distances between ZIP codes, phone locations; on-line tool)	\$50
	Linux/GNU/X by Yggdrasil Computing (1st production release; run from the CD; TCP/IP & NFS; drivers; MPEG; SCSI support; lots more)	\$45
	Walnut Creek C User's Group (Volumes 100 to 364)	\$40
NEW!	Walnut Creek Linux V0.99.13 (100's of tools; drivers for all sound, CDs, video; X-Windows; software engineering tools; easy install)	\$40
	InfoMagic Unix (three public domain Unix systems: 386BSD (version 0.1), Linux (version 0.99.10), and NetBSD)	\$40
	InfoMagic Source Code (Berkeley Net/2, MACH, GNU, Interviews, X, Andrew, XFree, Demacs & Winemac, djgpp, Modula-3, etc.)	\$40
	Knowledge Media Multimedia (625MB & 13,000 files; 1,232 sounds, 179 books, 100 movies, 114 stacks, 606 programs, 214 mods)	\$35
NEW!	Walnut Creek Space & Astronomy (1000 images, 5000 text files, sci.space archive, astronomy software; CD viewer included)	\$35
NEW!	Project Gutenberg (literature, historical documents, reference books, census data, religious documents, math constants, etc.)	\$35
	Knowledge Media Languages & Operating Systems (640MB of compilers, libraries, and operating systems; source code & executables)	\$35
	Walnut Creek X11R5 and GNU (X11R5 with contributed and comp.sources.x, over 120 GNU programs, complete C source)	\$35
	Walnut Creek Usenet and Simtel Unix-C (600MB)	\$35
	Walnut Creek Giga Games (arcade, simulations, card games, education, trivia, cheat sheets; some source)	\$35
	Austin Code Works Internet Warrior #1 (PC Internet tools: Gopher, Wais, Eudora, ph, Nupop, Trumpet, TCP/IP, FAQs, drivers, docs)	\$35
NEW!	NetGems of 1991 (comp.sources.*, alt.sources.* of 1991, plus X11R5 and GNUware and RFCs; Unix file system not ISO-9660)	\$20
	Walnut Creek Simtel 20 MSDOS Archive (C source code but lots of other stuff too)	\$20
	Sprite Network Operating System (source code & documentation of Ousterhout's Sprite O/S; Sun and DECStation boot images)	\$20

The Austin Code Works
11100 Leafwood Lane
Austin, Texas 78750-9587 USA

much more ... ask for catalog

Voice: (512) 258-0785
FAX: (512) 258-1342
E-mail: info@acw.com

Free surface shipping for cash in advance

For delivery in Texas add 7%

MasterCard/VISA

Listing 1 *continued*

```

output+=SKIP;
/* look for non-zero values and include in our COA calc */
if( val ) {
    denominator += val;
    /* ROM based later */
    numerator += val *(long)k;
}

}

/* divide if non-zero x/D */
if( denominator ) {
    /* return crisp value */
    /* Could be converted from float if desired */
    numerator = (long)((float)(numerator/denominator) * pwm_members.slope +
pwm_members.intercept);
    return( numerator );
} else
    return( 0 );
}

/* end COA calc */

/*****
Routine: fuzzify
Date: 4/3/93
Author: Jack J. McCauley
fuzzifier for our servo
*****/
int fuzzify( int derr_dt, int err,
            int *p_Terror, int *p_dTerror,
            int *p_out, int *resultant ) {

    static int k, val, alpha_cut, temp, *moe;

    /* COA method */

    /*
    Get the alpha cut for error and derivative. Use the AND (min) operator and cut the
    output, a non-zero fuzzy MIN indicates a rule has fired, write the cut to the result
    array by "shadowing" the existing using the MAX operator
    */

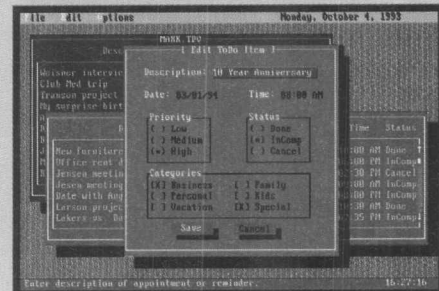
    /* normalize the output derivative */
    moe = resultant;

    /* Find out if the rule fired. If the rule fired then get the alpha cut */
    if( (alpha_cut = FUZ_AND( p_dTerror[derr_dt], p_Terror[err] )) != 0 ) {
        for ( k=0; k<ARRAY_SIZE; k+=SKIP ) {
            /*
            An interesting effect will be noticed if the skip is set greater than
            1. In this system setting skip to lets say 2 or three will yield
            in most circumstance yield the same defuzzified crisp output if the
            slope of the membership functions are not too steep.
            The other benefit is that the execution speed is increased greatly
            */

            /* create shadow */
            val = *p_out;
            /* don't get bit by shortcuts */
            val = FUZ_MIN( alpha_cut, val );
            temp = *resultant;
            *resultant = FUZ_MAX( temp, val );
            resultant+=SKIP;
            p_out+=SKIP;
        }
        /* rule fired */
        return( 1 );
    } else
        /* rule didn't fire */
        return( 0 );
}

```

Add a powerful user interface to your C/C++ apps with Vitamin C 4.0



The original user interface tool kit, Vitamin C's library of C functions makes your C/C++ programs faster to code and easier to use.

Simple C function calls get you up and running quickly with solid, reliable solutions including:

- ♦ Windows
- ♦ Menus
- ♦ Entry fields
- ♦ Validation
- ♦ Push buttons
- ♦ Radio buttons
- ♦ Check boxes
- ♦ List boxes
- ♦ Scroll bars
- ♦ Text Editors
- ♦ Help system
- ♦ List manager
- ♦ Full mouse support

Vitamin C's event driven, message based design gives you maximum flexibility to express your style and vision--not ours.

Of course, library source is included and your programs are royalty free!

Vitamin C 4.0

\$395.00

The next generation in C source code formatting, **CodeScan** makes any C source file easier to read, maintain, and debug.

With its unique, interactive control panel, you'll quickly and easily tell **CodeScan** how to format your code. No cryptic command lines or config files.

CodeScan 2.0

\$79.00

Order and information hotline

800-726-6447

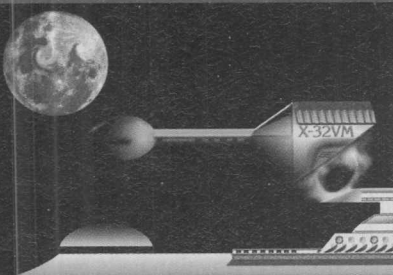
Creative Programming

P.O. Box 112097
Carrollton, Texas 75011-2097 USA
(214) 245-9139 Fax: (214) 245-9717

♦ Request 123 on Reader Service Card ♦

Flash Graphics

- Real & protected mode 16-bit support
- 32-bit protected mode support
- Wide range of graphic adapter support
- Virtual screens
- Multiple simultaneous graphics adapters
- BGI interface available

\$250, No Royalties**X-32VM 32-bit DOS Extender**


- Up to 3.5 gigabytes of virtual memory
- Ultra compact, self-contained executables
- Fully compatible with Zortech's DOSX
- Support for Watcom C9.5/386
- Support for Borland C++ for OS-2

\$250, No Royalties**Flash View**

- Source level debugger for C++, C and assembler programs.
- Easy to use expansion of classes, structures, pointers, arrays, etc.
- Powerful run-time memory protection detects pointer bugs
- Full nested call traceback display with automatics and parameters
- Unlimited number of break and trace points
- History buffer for recording up to 32,000 source lines and variables
- Program execution time recording for timing analyses
- Dual monitor debugging

\$250

FlashTek is proud to offer former Zortech products, by the original authors, now with support for other compilers. New features, fantastic support, no royalties, and great prices.

ORDERS 800-397-7310 

FlashTek, Inc. 121 Sweet Ave. Moscow, Idaho 83843 Info: 208-882-6893 Email: flash@proto.com FAX: 208-882-7275	FlashTek, Ltd. 3 Partnership House Witham Brook Park Grantham, Lines England NG31 9HZ Voice+44-476-74108 FAX+44-476-61382
--	---

Borland, BGI, Watcom and Zortech are trademarks of their respective companies.

◆ Request 276 on Reader Service Card ◆

Listing 1 *continued*

```

}
/* end FUZZIFICATION calc */

/*****
Routine: servo_torque
Date: 4/3/93
Author: Jack J. McCauley
fuzzy servos to torque set point
*****/
/*
Routine would run as an ISR attached to a timer interrupt passed values are read
from and A/D convertor and command torque (serial port etc..)
*/
long servo_torque( int error, int derr_dt )
{
    /* statics for speed */
    static int row, column;

    /* pointers to tables */
    static int *p_out, *p_dTerr_dt, *p_Terror;

    /* COA container class */
    static int resultant[ARRAY_SIZE];

    /* zero fill the array */
    for( row = 0; row<ARRAY_SIZE; row++ )
        resultant[row] = 0;

    /* normalizing the error derivative will allow the error to ff
    /* normalize error derivative to look up table RADIX*/
    error = (long)((float)error * torq_members.slope + torq_members.intercept);

    /* normalize error derivative*/
    derr_dt = (long)((float)derr_dt*deri_members.slope + deri_members.intercept);

    /* MAX and MIN error */
    if( error > MAX_ERROR )
        error = MAX_ERROR;
    else if( error < MIN_ERROR )
        error = MIN_ERROR;

    /* MAX and MIN derivative */
    if( derr_dt > MAX_DERI )
        derr_dt = MAX_DERI;
    else if( derr_dt < MIN_DERI )
        derr_dt = MIN_DERI;

    /* traverse the rule table and evaluate the rules */
    for( row = 0; row < DER_MEMBERS; row++ ) {

        /* get pointers to tables from data structures */
        p_dTerr_dt = dTerr_dt[row];
        for( column = 0; column < TORQUE_MEMBERS; column++ ) {

            /* get pointers to tables from data structures */
            /* get the output membership function for that rule evaluation */
            p_out = rule[row].table[column];
            p_Terror = Terror[column];
            /* fuzzify the rule */
            fuzzify( derr_dt, error, p_Terror, p_dTerr_dt, p_out, resultant );

        }

    }

    /* defuzzify using COA */
    return( defuzzify_COA( resultant ) );

    /* return the fired rules */

}

/* END */

```

Run-Time Type Checking in C++

Run-Time Typing

I have taught myself C++ and am comfortable with most of its concepts. There is one aspect that I am unsure about. Bjarne Stroustrup, in his book *The C++ Programming Language* states: "using run-time type inquiries ... destroys all modularity in a program and negates the aims of object-oriented programming." I attempt to adhere to this suggestion, but there is a situation in which I don't know how to apply the rule: reading and writing objects to data files. As an example, let's suppose we are writing a simple checkbook balancing program. The base object is an *Entry*, which corresponds to something you would enter in your checkbook. Derived from base class *Entry* are three classes which can be instantiated: *Check* (a written bank draft), *Deposit* (a bank window deposit), and *Withdrawal* (a window withdrawal). Keeping track of the type of objects in memory is easy, since when one of the three derived classes is created, it is done through a constructor which builds in type information. There are many ways to write the checkbook entries to a data file, so let's assume we just write a character-based representation to the file. The question is how can we write the data so that we can read the information back into memory? The only way I can think of is to include type information, such as an integer coded for each class type. This method violates Mr. Stroustrup's rule, though. Any thoughts you have on this matter would be greatly appreciated.

Paul Waldo
Forest, VA

A One reason Bjarne was against run-time checking was because it enables programmers to avoid derivation. For example, suppose you have a *type_of* function that can identify the *Entry* type of a pointer. To post an entry, you might code some-

thing that looks like Listing 1. If you need to add an additional type of *Entry*, then you have to add another case to the switch statement.

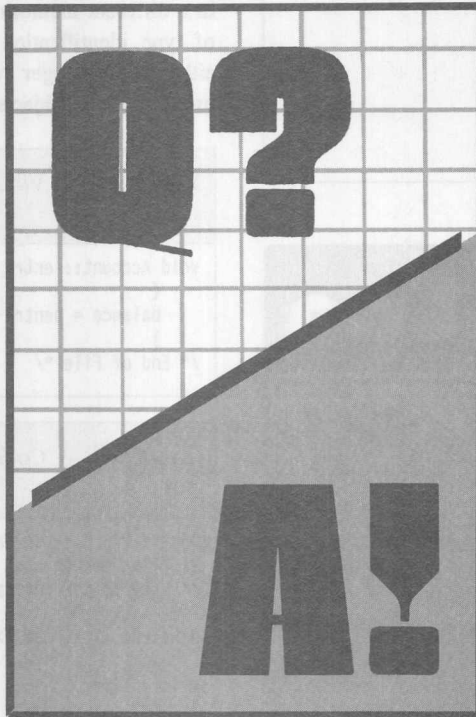
However, there's a cleaner way to handle this problem. Suppose you give *Entry* a pure virtual *post* function. Each class derived from *Entry* now supplies its own *post* function. The *post_entry* function could look like Listing 2. When adding a new derived class, you do not need to change the *post_entry* function.

You just provide a *post* function for the new class.

As you have suggested, virtual functions only work while objects are in memory. Each object of a class with virtual functions contains a pointer to a table of function pointers (the vtable, as it is sometimes referred to). All objects of a particular class contain a pointer to the same table. When the program calls a virtual member function it uses the pointer to the corresponding function in the vtable. In a sense, this vtable pointer uniquely identifies the class type of an object. In fact, some compilers have non-standard extensions that can use this pointer to provide a form of run-time type identification. Other vendors provide alternative methods for run-time identification. Microsoft has a *CRunTimeClass* object associated with each class that is used with the *IsKindOf* function. With this function, you can determine if an object belongs to a particular class or if it is derived from a class. (The class must be derived from the Microsoft *COb-*

ject class to work with *IsKindOf*.) To use the *IsKindOf* function, you include a *DECLARE_DYNAMIC* macro in the class definition and an *IMPLEMENT_DYNAMIC* macro in the class implementation. *CRunTimeClass* is the data type used to store class information. You can obtain a pointer to an object of this type with:

```
CRunTimeClass * pclass = RUNTIME_CLASS(Your_class);
```



Kenneth Pugh, a principal in Pugh-Killeen Associates, teaches C and C++ language courses for corporations. He is the author of *All On C*, *C for COBOL Programmers*, and *UNIX for MS-DOS Users*, and was a member of the ANSI C committee. He also does custom C/C++ programming and provides SystemArchitectonics™ services. His address is 4201 University Dr., Suite 102, Durham, NC 27707. You may fax questions for Ken to (919) 489-5239. Ken also receives email at kpugh@allen.com (Internet) and on Compuserve 70125,1142.

Typically you do not use the information in *CRuntimeClass* directly, but instead pass it to *IsKindOf*. For example, as shown in the following code fragment, you might want to cast a base class pointer to a pointer to a real object. To be logically correct, you need to be sure the object pointed to belongs to a particular class.

```
CObject * pyour_object = new Your_class;
...
if ( pyour_object->IsKindOf( RUNTIME_CLASS(Your_class) ) )
{
    // Cast it
```

Listing 1 One way to do run-time type checking

```
void Account::entry_post(Entry * pentry)
{
    switch(type_of(pentry))
    {
        case Deposit_entry:
            balance += pentry->amount;
            break;
        case Check_entry:
            balance -= pentry->amount;
            break;
        case Withdrawal_entry:
            balance -= pentry->amount;
            break;
    }
}
/* End of File */
```

```
Your_class * p_this_object =
    (Your_class *) pyour_object;
}
```

You can use this type information to create objects of a given class. The *CRuntimeClass* class provides a member function *CreateObject* for dynamically creating objects. The use of *CreateObject* is demonstrated as follows:

```
CRuntimeClass * p_your_class = RUNTIME_CLASS(Your_class);
// Create it
CObject * pyour_object =
    p_your_class->CreateObject();
// Cast it to the class
Your_class * p_this_object =
    (Your_class *) pyour_object;
```

In your question, you have run across one area of programming that requires some form of type identification: permanent or persistent storage. You cannot use the address of a vtable as the identification means for an object. When the contents of an object are read back into memory, the chances are that the vtable will be in a different memory location. You will need to store some form of type identification with the object. This identification could either be an integer value or the string name of the class. If your program stores objects in a known sequence and retrieves them by

Listing 2 Virtual functions simplify run-time type checking

```
void Account:: entry_post(Entry * pentry)
{
    balance = pentry->post(balance);
}
/* End of File */
```

Listing 3 Code that one compiler didn't like

```
/* GRM-P45.C -- Graham, Learning C++, p. 45 */
// File rabbits.ccp
// Program for inverse Fibonacci rabbit problem
```

```
#include <iostream.h>
```

```
main()
{
    // Get input from user

    long current; // number of pairs this month
    long fertile; // number of fertile pairs
    long needed; // number of pairs needed
    ...
}
```

Error messages:

```
Compiling C:\GRM-P45.C:
Error E:\VC\INCLUDE\IOSTREAM.H 38: Declaration syntax error
Error E:\VC\INCLUDE\IOSTREAM.H 39: Declaration syntax error
Error E:\VC\INCLUDE\IOSTREAM.H 42: Declaration syntax error
...
// End of File
```

Little Big Lan

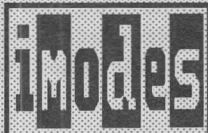
◆ Peer to Peer LAN, to 250 nodes

◆ \$75 total software cost!
No matter how many nodes!

◆ Use Ethernet, Arcnet or serial/parallel/modem to connect.

◆ Mixed mode routing
Any combination of above connection is possible on any given node.

◆ Use with windows
Seen as 100% compatible Microsoft Network by Windows 3.x.



Information Modes

P.O. Drawer F
Denton TX 76202
817-387-3339 Techline
817-382-7407 Fax

1-800-628-7992 Orders

The \$25 Network

Try the 1st truly low cost LAN

- Connect 2 or 3 PCs, XTs, ATs, PS/2s
- Uses serial ports and 5 wire cable
- Runs at 115K baud, up to 90 feet
- Transfer 8500 bytes per second (ATs)
- Runs in background, totally transparent
- Share any device, any file, any time
- Needs only 15K of ram
- Just \$25 per network, NOT PER NODE!
- Replace all file transfer software
- Version 2.3P now has TinyMAIL
- OVER 20,000 SOLD WORLDWIDE

Skeptical? We make believers!



the same known sequence, then you do not need any type identification stored with the object. For example, if you always store *Checks* starting at the first position in the file, followed by *Deposits* and then *Withdrawals*, you can determine the type of an object by its position in the file. In your example, you cannot assume this kind of ordering exists, so you *must* store a class identifier. There are lots of ways to store this identifier, depending on how your classes are organized. Let's assume you have some unique identifier for each account entry type, such as:

```
enum Entry_type {Entry_check, Entry_deposit,
    Entry_withdrawal};
```

One of these values would be stored away with each object. For example, if your compiler provides run-time type identification with *type_of*, you might code:

```
Entry::save()
{
    if ( this->type_of() == Check )
        // Store Entry_check
    else if (this->type_of() ==
        Deposit )
        // Store Entry_deposit
    else if (this->type_of() ==
        Withdrawal )
        // Store Entry_withdrawal)
    ...
    // Store remaining contents
    // in an account
}
```

Before I get too much mail regarding this "hidden switch statement," let me explain that this function is complementary to the retrieval function, which I will show shortly. You could use a virtual function for *save* in each of the derived classes. The function would store the appropriate *Entry_type* value, call a *save* function in the *Entry* class to store the data members in that class, and then save its own data members. If you were using the Microsoft compiler, this type checking code might look like the following fragment. However, Microsoft provides other functions that make this code unnecessary, as we shall see shortly.

```
Entry::save()
{
    CRuntimeClass * pcheck_class
        = RUNTIME_CLASS(Check);
    CRuntimeClass * pdeposit_class
        = RUNTIME_CLASS(Deposit);
    CRuntimeClass * pwithdrawal_class
        = RUNTIME_CLASS(Withdrawal);
```

```
if ( this->IsKindOf(pcheck) )
    // Store Entry_check
else if (this->IsKindOf(pdeposit) )
    // Store Entry_deposit
else if (this->IsKindOf(pwithdrawal) )
    // Store Entry_withdrawal)
...
// Store remaining contents in an account
}
```

C-Index/II

Data Management Library

Discover Why

Leading software developers choose C-Index/II for their advanced retail products and in-house applications.

C-Index/II: \$695.00

- Fast B+Tree Indexing
- Unlimited File Size and Key Types
- Complete Portable C Source Code
- Any system with ANSI or K&R C
- Nine Simple High-Level Functions
- Over 60 Additional Functions
- Multi-User and Single-User Access
- Data and Indexes in Same File
- Multiple Record Formats per File
- No Application Royalties

C-Index/PC: \$195.00

- Most of the features of C-Index/II
- 32 Megabyte Files and String Keys
- Complete C Source for MSDOS

New Release 4.1

PowerFail Protection
Transaction Logging
File Mirroring
Fast Sequential Access
Multiple B+Trees / File
Windows NT Port

Windows 3.x & Windows NT
MSDOS UNIX PharLap
Microsoft & Borland C/C++

Call or Fax Today for Your Free Demo Kit



Trio Systems

936 E. Green St. Suite 105
Pasadena, CA 91106
818/584-9706
818/584-0364 Fax

The basic dilemma comes in retrieving the objects. You cannot use a *retrieve* function for each derived class, as you do not know the type of entry for the next object stored in the file. Thus you can only retrieve an *entry* as a base class object. For example, the caller might use something that looks like:

```
Entry *pentry;
Account account;
...
account.retrieve_next(&pentry);
```

The *retrieve_next* function could look like the following:

```
int Account::retrieve_next(Entry **pentry_in)
{
    // Get rid of old pointer
    Entry *pentry = *pentry_in;
    if (*pentry != NULL)
        delete pentry;
    // Read the record off disk
    // Then check the type of the record ready
    if (type == Entry_check)
        *pentry = new Check;
    else if (type == Entry_deposit)
        *pentry = new Deposit;
    else if (type == Entry_withdrawal)
        *pentry = new Withdrawal;
```

```
// Move associated data into the type
...
// Then return the pointer
*pentry_in = pentry;
}
```

The caller passes this function a pointer to pointer to the *Entry* class. The function deletes the *Entry* that was pointed to by *pentry_in*. The *Entry* class should provide a virtual destructor, so that any additional data members in the derived classes are deallocated. Based on the *Entry_type* value stored in the file, the function allocates the appropriate pointer. The function moves the necessary data from the file into the new object and returns the value of the pointer. If necessary, each derived class can include a function that reads any additional data members from the file.

The Microsoft archive retrieval function (*CArchive::operator>>*) does not require this if-else structure. The storage function (*CArchive::operator<<*) stores a run-time class identifier (the name of the class) in the file. When the retrieval function reads the file, it dynamically creates an object of the stored class and loads the information from the file into that object. Microsoft's multi-pronged approach to run-time typing eliminates worries about the details of persistent object storage and retrieval. All you have to do is to include the necessary macros in your object header file and your object implementation file.

C++ Problems

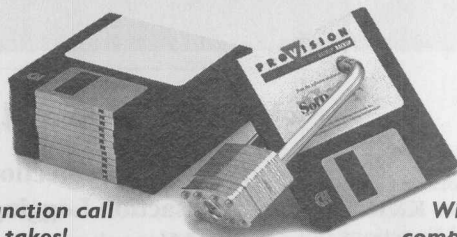
I am an agricultural economist at the U.S. Department of Agriculture and am trying to learn C and C++. I have bought over ten books on C, and am reading them and keying in the exercises to practice the concepts. One of the books I have bought is *Learning C++*, by Neil Graham. I began keying in one of the exercises (in C++) and the program gave me a bunch of *iostream* declaration errors when I attempted to compile the source code using Borland Turbo C++, 1991 (Listing 3). I saw your column in *The C Users Journal* and thought that you might be willing to suggest what is wrong, and how I might fix it. Also, do you have any recommendations for textbooks and/or diskette tutorials that may be useful in learning C and C++?

Kenneth W. Erickson
Washington, D.C.

A Reading several books and trying their examples is an excellent way to learn a new language. You can get a good feel for alternative ways of approaching problems. However, when you use only books for learning, you often run into problems for which a book has no answer. Many times I've been in the same quandary when using vendor-supplied manuals. In some cases the solution to my problem was simple but the manual just didn't deal with the problem.

In your case, what appears to be an unexplainable error is caused by a simple glitch. You named the program with a ".c" extension. The compiler took this to mean that your program was to be compiled as a C program. Inside the *iostream.h* file are several C++ syntactic constructs (such as *class*). The compiler reported error messages for these constructs since they do not exist in C. If you had named the program with a .cpp extension, the compiler would have cleanly compiled the program. You might have been confused if you compiled other C++ programs without

ProVision:Backup Saves the Day



**One function call
is all it takes!**

**Windows
compatible!**

Now you can save yourself unnecessary programming time, and save your client's valuable data, too. With ProVision:Backup, you can build a complete backup and restore system into your C applications with as little as one function call! It's the convenient library that takes away the excuses – and blame – for not backing up. Complete with handy features like: Automatic on-the-fly formatting; optional compression during backup procedure; optional detection when a diskette is inserted, works with all floppy types; complete configurability... you can provide the user interface. It's fast... more than twice the speed of DOS BACKUP and not DOS-version dependent. Save your time while saving your client's data with ProVision:Backup.

\$199 (US)
plus shipping

PROVISION™
BACKUP BACKUP BACKUP

FREE
runtime
distribution.

For ordering and information call toll free

1-800-755-7344 (U.S. and
Canada)
Major credit cards accepted

from the software architects at

SofDesign
INTERNATIONAL, INC.

SofDesign International, Inc.
1303 Columbia Dr., Suite 209 Richardson, TX 75081 USA
Voice: (214)644-0098 • FAX: (214)644-4286

© 1994 SofDesign International, Inc.

◆ Request 215 on Reader Service Card ◆

errors and therefore you thought there was something wrong with this program. Many compilers (such as Borland), have a switch that compiles .c files as C++ programs. If this option was set in your configuration file for previous programs, then other C++ programs with .c extensions would have compiled properly. If you switched around directories or reinstalled the compiler, the switch may have been reset.

There are numerous books out on C. Many are based on your knowledge of other languages. My book *C Language for Programmers* (QED) gives comparisons of C constructs with COBOL, PASCAL, PL/1, BASIC, and FORTRAN. My new book *C Language for COBOL Programmers* (QED) presents very detailed comparisons between COBOL language statements and C. *All on C* (Harper Collins) explains C without comparisons to other languages, but with numerous examples. In the C++ arena, I suggest *C++ Programming and Fundamental Concepts* by Anderson and Heinze (Prentice-Hall). I use that book as a supplementary book for my C++ courses.

User interface

Q I am new to the C language and am having a problem that you might help me with. I am writing a football card data base, and need help with the user interface. What I am trying to do is let the user of my program enter required information without pressing the enter key. I also want the user to be able to move from one data entry field to another using the arrow and home keys of the key pad. If the user presses the arrow keys the highlighted field would move up or down as required, but if a letter key or a number key was pressed, the letter or number would be concatenated to the appropriate char string variable. The concatenating part I can handle. It's getting the input that's giving me the problem. I've tried many different approaches to make this work, but still no luck. If you could help me with this problem I would be very grateful. I'm a subscriber to the *C Users Journal*, and will be checking the Questions & Answers section to see if you can solve my problem. I am using Borland C++ version 3.0 (DOS).

Randy Jones
Ruther Glen, VA

A As you have discovered, data-field entry functions are not included in the Standard C libraries. Numerous shareware and commercially-available libraries will perform the operations you list. For a listing of available shareware consult *The C User's Group Public Domain Catalog* [available for free upon request from R&D Publications, 1601 W. 23rd St., Lawrence, KS 66046. Ph: (913)-841-1631. Fax: (913)-841-2624. e-mail: cuser@rdpub.com. Perusing *The C Users' Journal* itself will reveal a number of the major vendors of display packages. Since for the past year I have been programming almost exclusively in Microsoft Windows using Visual C++, I haven't kept up with all the different features of the commercial packages. Many of these packages have integrated screen designers/code generators. With such a system, you can layout your screens with a mouse or cursor keys, rather than by writing individual function calls for each field. In case you can't find what you want in either the catalog or the ads, my book *All on C* shows a sample implementation of a package of display functions providing most of the features you requested. □

Innovative Developer's Tools!

TE Developer's Kit

Incorporate text editing features into your application easily and cost effectively. Features: multiple files/windows, word-wrap, edits large files, reconfigurable keyboard and screen colors, undo, cut/paste, printing, and search/replace. The word processing features include **bold**, underline, and *italic* styles. Paragraph features: **indentation, double spacing, centering, and justification**. The Windows and OS2 versions also include **multiple fonts and point sizes, imbedded pictures**, and many more character styles.

Additional features for the Windows version: ruler and tab stops, page break and automatic pagination, DLL and custom control interface, print view edit mode for **Wysiwyg** editing, RTF support for input/output and clipboard, hanging indents, multiple columns and more. All versions include the complete 'C' source code. (DOS: \$299, Windows: \$349, OS2-PM: \$379)

ReportEase: Report Writer/Mail Merge Engine

ReportEase consists of a report layout editor and a report executor. Advanced features include: multiple files, multiple sorts; data, calculation, system, and dialog fields; bold, underline, italic formats; subtotals, record filter, functions, word wrapping and more. Simple interface works with any application. Includes the 'C' source code. (DOS: \$349, Windows: \$379)

Also available **ReportEase Plus** for Windows featuring a **graphic form editor with line/box drawing, colors/shades, drag/drop, print preview**, etc. (\$419)

Spell Time

Spell Time consists of a dictionary (over 100K words) and the routines to access the dictionary. It also includes an application specific dictionary, and a user dictionary. The routine will suggest alternatives for a misspelled word. Highly optimized: 400 to 500 words/sec on a 33 Mhz 386 computer. An interface with TE included. Includes the complete 'C' source code. Specify DOS, Windows or OS2-PM. (\$369)

ChartPro

This Windows' DLL draws bar, pie, line, area, xyz point chart, and hilo presentation graphs in unlimited styles. The output can be sent to a window, printer or a bitmap. Features 3d rotation and dialog boxes to edit chart parameters. Includes the 'C' source code. (\$379).

MultiSpawn

MultiSpawn offers *multiple* methods of running large programs by swapping out the parent program. Restores the complete status of the parent program. Royalty Free. (\$199)

Sub Systems, Inc.

1-800-447-6819

Fax: 1-617-438-0311

159 Main Street, #8C, Stoneham, MA 02180, (617)-438-8901

✦ Request 452 on Reader Service Card ✦

Weight Reduction Techniques in C++

Randy Kamradt

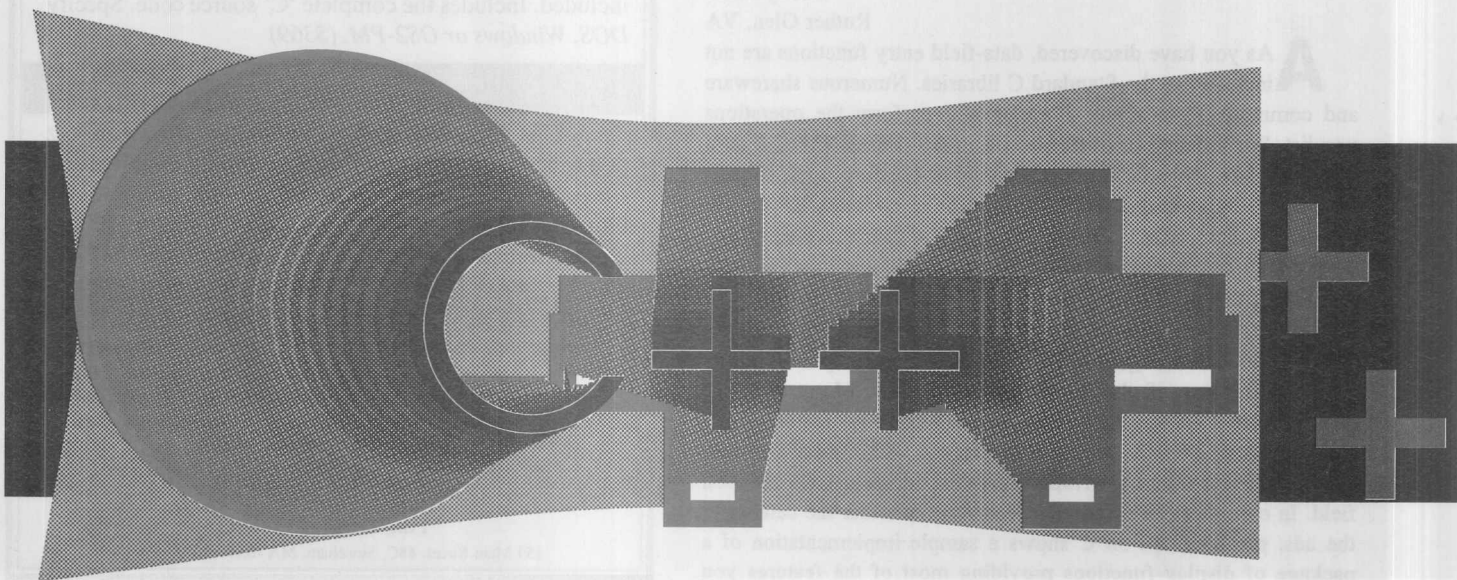
Introduction

I have been working on a large scale database downsizing project for the Ventura County School District since the beginning of 1993. Recently we turned over to the users the first application, a teaching-credential tracking database. Instead of the ticker tape parade we had expected, all they did was comment on how slow it seemed to run. Even though we used all the latest buzz words — such as “object-oriented” and “client-server” — we had to admit it was very sluggish. So we began an intense optimization effort, putting our application on a crash course of exercise and diet, to get it down to size and up to speed. This article deals with the potential problems facing object-oriented programming in C++ and some of the solutions we used.

One of the major advantages to object-oriented programming is that it can hide complexity. Very complicated sub-systems can be wrapped by classes. These classes provide well defined interactions, making their objects relatively immune to environmental factors. The down side to this is that the application programmer may not be aware of all the ramifications of using an object. Objects may encapsulate the acquisition of system resources, or they may create sub-objects. Just declaring a local variable can be a costly procedure.

One problem that faced us early on was from a class of the Commonbase Database class library, *DBTable*. Just creating a variable of type *DBTable* had the side effect of opening a communication channel (socket) to the database server. We were allocating *DBTable* objects from free store and in rare cases they never got deleted. Since our networking software had only 20 sockets available, the application would run fine for a while and then run out of sockets. This is actually an old problem, familiar to C programmers, but with a new twist: more than memory might be allocated by creating a user-defined type. Since this caused the program to die it was fixed early on, but other similar problems existed. We had to automate the creation of each *DBTable*, so as not to create it until it was needed, and to delete it as soon as possible. It could not be left up to the application programmer.

Another side of complexity hiding is that an object may be a front for many sub-objects. One of our objects, *BMOJoin* (Business Model Object Join), contained a list of *DBTables*. A *join* is database jargon for a relational combination of tables. Since a *DBTable* holds one socket, a *BMOJoin* can hold many. To add to this problem, another object, *BMOIterator* has a pointer to a *BMOJoin*. As I will explain later, the *BMOIterator* was the key to controlling all these resources, and minimizing their allocation.



Another class, *ACond*, represents a logical condition. It is implemented as an expression tree. So doing a simple assignment of an *ACond* could be costly. Again this is actually an old problem, the only difference being that in C such copying can be done only by calling a function, while in C++ the function can be initiated with an assignment operator. C programmers know that calling a function can take time and resources, but using built-in operators always takes (relatively small) constant time and never takes system resources. These assumptions go out the window in C++.

Free-store usage is another problem. Just opening a window and tying it to a database in our application takes thousands of memory allocations, some of them just for a few bytes. Memory allocation is the lifeblood of C++ programming because objects should be made autonomous, and cannot rely on outside sources such as local or global variables for their internal memory space.

A string class, for instance, typically contains a pointer to memory containing the string (see Figure 1). That memory must come from free store if the class is to be considered general purpose. If the memory came from outside the class, the string could not delete it when it was done, and the user would have to delete it if necessary. The issue of pointer ownership is important in making a safe application, and helps alleviate some of the common C pointer troubles such as deleting memory twice, or forgetting to delete it at all. It also means that memory allocation and copying may be done more often than absolutely necessary.

Another problem introduced with object-oriented programming is over-generalization. With base classes one can provide the common subset of functions available from a set of derived classes. If the programmer uses the base class, the full set of functions may not be available. An example of this is the use of a list object. A list object might be represented either by a linked list or by an

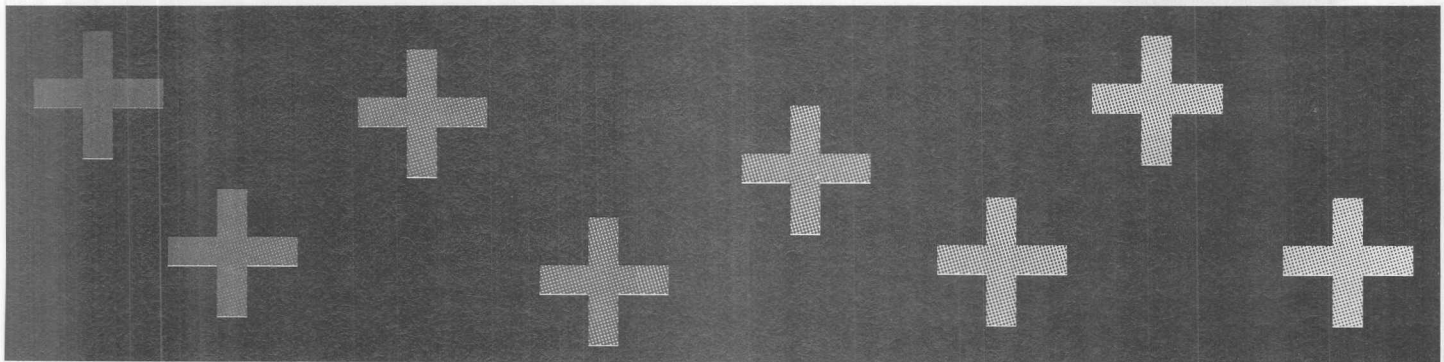
array. If the programmer uses a list and needs to get to the n -th object, indexing may not be an option. The programmer will have to iterate through the list, keeping a count. If the programmer uses an array object, indexing is available.

Another example of this is the use of a complex number class. The operation $(2 + 0i) * (6 + 0i)$ can be calculated much easier as $2 * 6$. Because the more general form is used, the optimization for a special case is missed. The ability to generalize is a major plus for C++, but can limit optimization.

Code Bloat

Another issue I deal with here is code bloat. I used to think a program was big if I had to leave small model (greater than 64 kilobytes of code or data). Our current application is about 1.5 megabytes and growing (from about 30,000 lines of code and two libraries). Part of this growth stems from the tendency to make objects all-purpose (or worse, multi-purpose). Poorly designed objects often give liberal access to their internals, or multiple methods of access. A well designed object should do one thing, one way, and do it well. The public interface should be minimal in order to guide the programmer to the expected usage, and to keep things simple.

Making matters worse, currently the linkers I use (Borland's *tlink* and HP-UX *ld*) always link in all virtual functions even if they are never used. This is because all virtual functions are referenced in an object's virtual table, which is included with a class constructor. I have heard that new linkers are addressing this problem, but without thorough evaluation of all the code, they will have trouble determining whether any one virtual function will assuredly be called (or assuredly *not* be called).



Randy Kamradt has been programming in C/C++ for the past seven years. He is currently working for TEAM Software, a fashionably lean consulting company developing an integrated database package for the Ventura County Superintendent of Schools, in Ventura California.

Figure 1 *Good objects must do their own memory management*

```

class BadString {
public:
    // Using the pointer directly:
    BadString(char *s) { ptr = s; }
    // Where did pointer come from:
    ~BadString() { delete[] ptr; }
private:
    char *ptr;
};

char *strdup(const char *str)
{
    return strcpy(new
        char[strlen(str)+1],str);
}

class GoodString {
public:
    // Don't use the pointer, copy it:
    GoodString(const char *s)
    { ptr = strdup(s); }
    // Now you can safely delete it:
    ~GoodString()
    { delete[] ptr; }
private:
    char *ptr;
};

// End of File

```

Another culprit in code bloat is the inline function. In the C++ style of object-oriented programming, small functions are very prevalent, and inline functions are necessary for a well oiled program. The factor that should decide whether a function is inline is the ratio of time spent calling the function to the time spent inside the function. If the time spent inside the function is much greater than the time spent calling the function, making it inline

Figure 2 *Using the profiler to find hidden problems*

```

// Why isn't the while loop
// ever executed?

time    count
0.0012  50   int flag = 0;
0.0031  50   while(flag = 0)
           {
0.0000   0       if(doTilTrue())
0.0000   0       flag = 1;
           }

// End of File

```

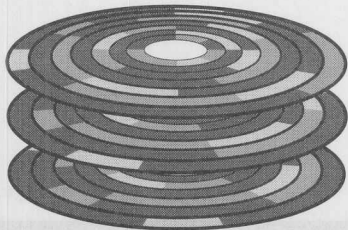
won't help speed much and will simply contribute to excessive code size. This implies that to use inlines effectively one should be aware of the inner workings of the compiler, and all of the side effects caused by some C++ functions.

One of the best ways to optimize is to use a profiler. I used the Borland Windows profiler extensively while optimizing. In spite of its tendency to crash, or reboot my computer occasionally, it was a tremendous help not only in optimizing, but in finding bugs and memory leaks. The Borland profiler is interactive, much like a visual debugger. It allows you to stop the program at any point to examine statistics collected, to turn profiling on and off during a single run, and to selectively profile portions of the code. It provides time spent on a single line of code, and the number of times a single line is called. This can be very handy for finding hidden bugs (see Figure 2).

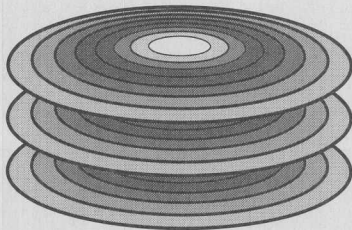
Profiling event-driven programs can be a challenge since the main loop of the program may be inaccessible. The method I

DISK_PAK™ for UNIX®

No More Fragmented Disks!



Fragmented disks before DISK_PAK



Optimized disks after DISK_PAK

- Defragments disks in-place
- Visually displays disk organization
- Improves disk performance
- X Windows or character terminals
- Available on SunOS®, Solaris®, SCO® UNIX, DG/UX®, and AIX®

Call for your free
Disk Fragmentation
Analysis Kit!
800-477-5432



EAGLE Software, Inc.
123 Indiana Avenue
P.O. Box 16
Salina, KS 67402-0016
Phone: (913) 823-7257
FAX: (913) 823-6185
e-mail: info@eaglesoft.com

DISK_PAK is a trademark of EAGLE Software, Inc. UNIX is a registered trademark of UNIX System Laboratories, Inc.

◆ Request 183 on Reader Service Card ◆

Listing 1 *Defines class MemoryPool*

```

#ifndef MEMPOOL_H
#define MEMPOOL_H

#include <stddef.h>

const CharSize = 8;
const PoolSize =
    sizeof(unsigned long)*CharSize;

class MemoryPoolLink {
private:
    friend class MemoryPool;
    MemoryPoolLink(size_t _size,
        MemoryPoolLink *_next);
    ~MemoryPoolLink();
    void *malloc(size_t size);
    void free(void *, size_t size);
    unsigned long bits;
    MemoryPoolLink *next;
    char *data;
};

class MemoryPool {
    MemoryPoolLink freeHead;
    MemoryPoolLink usedHead;
    size_t size;
public:
    MemoryPool(size_t size);
    void *add();
    void *malloc();
    void free(void *);
};

#endif

```


used was to profile each of the main classes of the program individually. Since we keep all of the code for a class in one module, that meant compiling that module with the debugging option on. In the profiler I could then set a profile area on every line in the module. After running the program and doing a few transactions that would exercise the class I was profiling, the statistics window would show me where the most time was spent. I used this technique for finding candidates for inlining and other optimizations. By profiling the main classes first, I was able to tell which of the sub-classes were in need of optimizing, focusing my effort where it was needed.

Figure 3 *Inlining virtual functions*

```
class Base {
public:
    virtual void print()
    { printf("Base"); }
};

class Derived1 : public Base {
public:
    virtual void print()
    {
        // Base::print() can be inlined
        Base::print();
        printf("Derived1");
    }
};

class Derived2 : public Base {
public:
    virtual void print()
    {
        // Base::print() can be inlined
        Base::print();
        printf("Derived2");
    }
};

void function(Base *bp)
{
    // Doesn't know whether
    // to call Derived1::print()
    // or Derived2::print(),
    // won't inline:
    bp->print();

    // Forced to call Base::print(),
    // can inline:
    bp->Base::print();
    Derived1 d1o;
    // Forced to call
    // Derived1::print(), can
    // inline:
    d1o.print();
}

// End of File
```

There are some limitations on inlining virtual functions. If a virtual function is called via a pointer or reference to an object, the actual function that gets called depends on the original type of that object. The compiler cannot determine at compile time the correct function, and therefore it cannot inline it. Virtual functions that are called via an actual object, or ones that are explicitly called with the `::` operator, can be inlined (see Figure 3). In at least one case we changed a virtual function to a non-virtual function in order to take advantage of inlining. This step must be taken with caution, however, possibly adjusting for any change in functionality.

The decision to inline doesn't need to be all or nothing for a function. There might be a situation where a function can be split apart to facilitate inlining. Figure 4 shows that the member function `GetWidgetPointer` gets called 10,000 times, but only has to create a `Widget` 10 times. By splitting this function in two, the part that gets executed 10,000 times can be inlined, and the main part of the code can be isolated in a private member function.

Some special precautions should be taken when inlining constructors and destructors. The compiler may add code that you didn't call explicitly. For example with an inline constructor, the compiler will also inline for you the setting of the virtual table pointer, calls to any base class constructors, and calls to constructors of any data members that have constructors. Borland C++ also adds code to check if the constructor is called on the behalf of a `new` statement and optionally calls a memory allocation routine. The destructors will do the same.

One more precaution for inlines, if the inline contains a function call, that function call may also be inline, which may also contain an inline function, and so on. Make sure you know what you put inline. Just as an aside, some current compilers forget to call the destructors for local variables in inline functions. This is important if the local variables hold memory or system resources and can cause a memory leak. For now I wouldn't create local variables or pass-by-value user-defined objects in an inline function.

OPTLINK[®] 5.0 for Windows

Now shrinks program file sizes another 50%!

Enables Windows developers to generate compressed self-loading executables (.EXE) and Dynamic Link Libraries (.DLL), thus reducing file sizes an additional 50%.

Benefits include:

- Loads applications faster
- Saves you \$'s - fewer diskettes to ship
- Speeds up installation time
- Complicates reverse engineering
- Reduces your customers' disk requirements
- Only requires relinking

Other reasons to use OPTLINK:

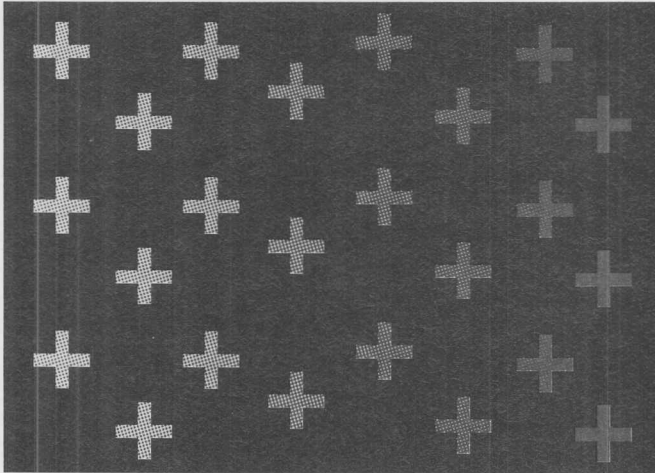
- Fastest Windows linking
- Unrivalled capacity
- Builds the smallest programs
- No limits on debug information
- Eliminates RC, IMPLIB, IMPDEF

OPTLINK 5.0 also offers several ease-of-use and capacity enhancements focusing primarily on the process of importing and exporting from DLLs. It now eliminates of capacity problems inherent in exporting C++ mangled names. Supports Borland, Symantec, and Microsoft C/C++. Full debug support for CodeView and Turbo Debugger. 30 Day Money Back Guarantee.



SLR Systems, Inc.
1622 N. Main Street, Butler, PA 16001 USA
Phone: (412) 282-0864; Fax: (412) 282-7965





Pointer Tricks

One of the best ways to address rampant free-store allocation is by counting pointers. This is a simple but effective technique that can be used on any class to speed up assignments and passing by value. It is best targeted at utility classes that are used in many different places and cannot be isolated.

A good example is a string class. I pointed out above that a class always allocating its own memory can lead to excessive memory allocation and copying. But if the string class contains a pointer to an intermediate data structure with a count, assignment becomes a simple matter of decrementing and incrementing a counter (see Figure 5).

Graphics & Timing Tools

BGI SVGA Video Driver Toolkit

New Toolkit!

Provides Super VGA driver support at up to 1280x1024x256 for most popular SVGA cards using the Borland BGI graphics library. Support for graphics mode mouse and multiple video pages. Import/export popular bitmap file formats. Supports Borland DOS language compilers in real and protected mode. \$129.95 w/source.

BGI Font Toolkit

New Toolkit!

Complete replacement for the graphics text functions in the Borland BGI graphics library. Corrects bugs in BGI text functions, allows arbitrary rotation, bolding, underlining, and italicizing of BGI fonts. Supports additional bitmap font formats for filled font rendering. Works with any BGI video or hardcopy driver. Supports Borland DOS language compilers in real and protected mode. \$89.95 w/source.

BGI Printer Driver Toolkit

New Version!

Provides drivers for Borland's BGI graphics library to support a wide variety of printers, plotters, and bitmap file formats. Support for EMS/XMS. Print popular bitmap file formats. Extensive color device support. Not a screen dump - load our drivers with BGI's *initgraph* and get full output device resolution. Supports Borland DOS language compilers in real and protected mode. \$129.95 w/source.

BGI For Windows

Port DOS BGI Code to Windows

Gives you an interface to the Windows 3.x GDI compatible with Borland BGI graphics calls. Port your DOS BGI graphics code effortlessly to Windows. Full support for 256 color palettes, BGI stroke fonts, high resolution displays, and rasterized hardcopy. BGI extensions support TrueType and 24 bit color. Supports Borland Windows language compilers. \$129.95 w/source.

PC Timer Tools

Event Timing and Scheduling

Brings microsecond resolution timing to your DOS application with extensive functions for timers, delays, alarms, timer tick management, and thread scheduling. Ideal for execution profiling, data acquisition, and process control. Supports Borland DOS compilers, MSC/C++, Intel 386 CB, Zortech, \$89.95 w/source. PC Timer Objects for C++ and Turbo Pascal OOP - \$89.95 w/source.

All toolkits include

The Official Fine Print

complete source (in both C and Turbo Pascal), object/driver/DLL distribution license, and our 30 day "No Questions Asked" return policy. Add \$5 shipping USA, \$10 elsewhere. VISA, MasterCard, and American Express accepted. Our toolkit code is found in a wide variety of commercial and private applications in daily use by over 100,000 end users.

Ryle Design

Purveyors of Big Science since 1987

PO Box 22, Mt. Pleasant, MI 48804 USA

Voice/Fax: 517.773.0587 CIS: 73047,1765

Demos and spec sheets available on our BBS: 517.772.2393

◆ Request 110 on Reader Service Card ◆

By adding a little smarts to the string class we can reduce the amount of work done, but we don't have to change the external appearance of the class. It is important to maintain the external appearance if you need to change the class, but don't want to make changes to all modules that use the class.

One problem still remains. If you assign *string1* to *string2*, then modify *string1*, *string2* will also get modified. This problem is solved by using "copy-on-write" semantics. Copy-on-write is a strategy where any member function of the string class that modifies the string will first call a function that splits off a private copy of the internal implementation (see Figure 6).

In our application we use both pointer counting and pointer counting with copy on write. The class *BMOIterator* mentioned above uses pointer counting as a smart pointer. Besides constructors, assignment operators, and a destructor, the only member function is the overloaded *operator->*. An object used with this operator appears as simply a pointer (see Figure 7). In order to work intuitively as a pointer, we did not implement copy on write. Also, since a *BMOIterator* holds system resources, we judged it best that no copying should take place unless explicitly asked for.

In another class, *Attributelist*, which is basically a list of integers, we did implement copy on write. Since these *Attributelist*s are not often modified, adding copy on write doesn't add significantly to run time. However after running the *Attributelist* code

Figure 4 Efficient inlining

Figure 4a. Splitting functions to facilitate inlining.

```
5.1274 10000 Widget *WidgetHome::GetWidgetPointer()
{
1.0333 10000   if(widgetPtr == NULL)
{
0.0012 10       widgetPtr = new Widget;
0.0013 10       if(widgetPtr == NULL ||
0.0000 0         widgetPtr->isError())
0.0000 0         return NULL;
1.0237 10000   } return widgetPtr;
3.5429 10000 }
```

Figure 4b. Inlining only the part called most often.

```
// the inline doesn't show up on the profile count:
inline Widget *WidgetHome::GetWidgetPointer()
{
    return widgetPtr ?
        widgetPtr :
        PrivateGetWidgetPointer();
}

0.0020 10 Widget *WidgetHome::PrivateGetWidgetPointer();
{
0.0059 10     widgetPtr = new Widget;
0.0035 10     if(widgetPtr == NULL ||
0.0000 0       widgetPtr->isError())
0.0000 0       return NULL;
0.0008 10     return widgetPtr;
0.0001 10 }

// End of File
```

Listing 2 *Memory pool member functions*

```
#include "mempool.h"

MemoryPoolLink::
MemoryPoolLink(
    size_t size,
    MemoryPoolLink *_next)
: bits(0),
  data(new
    char[size*PoolSize]),
  next(_next)
{
;
}

MemoryPoolLink::
~MemoryPoolLink()
{
delete[] data;
}
```

Figure 5 *Counting pointers*

```
class String {
private:
    struct StringImp {
        char *ptr;
        unsigned count;
        StringImp(const char *str)
            : ptr(strdup(str)),
              count(1)
        {
;
        }
        ~StringImp()
        {
delete[] ptr;
        }
    } *imp;
public:
    String(const char *str)
    {
        imp = new StringImp(str);
    }
    String(const String &str)
    {
        imp = str.imp;
        imp->count++;
    }
    // assignment is a little tricky
    String &
    operator=(const String &str)
    {
        // increment first in case
        // of assignment to self
        str.imp->count++;
        // be sure to clean up the old imp!
        if(--imp->count == 0)
            delete imp;
        imp = str.imp;
        return *this;
    }
    ~String()
    {
        if(--imp->count == 0)
            delete imp;
    }
};
// End of File
```

Figure 6 *Implementing copy on write*

```
class String {
// the contents of the string
// class from Figure 5.
...
// this indexing operator could
// modify the string
char &operator[](int i)
{
    if(imp->count > 1)
        Split();
    return imp->ptr[i];
}
// This indexing operator won't
// modify, don't split.
char operator[](int i) const
{
    return imp->ptr[i];
}
private:
    void Split()
    {
        // Create private copy:
        imp->count--;
        imp = new
            StringImp(imp->ptr);
    }
};
// End of File
```

Figure 7 *Implementing smart pointers*

```
class BMOIteratorImp;

class BMOIterator {
public:
    BMOIterator(const char *dbname);
    BMOIterator(const BMOIterator &);
    BMOIterator &
    operator=(const BMOIterator &);
    ~BMOIterator();
    BMOIteratorImp *operator->()
    {
        return imp;
    }
private:
    BMOIteratorImp *imp;
}

function()
{
    BMOIterator it("DBNAME");
    // AddCol is a member of
    // BMOIteratorImp
    it->AddCol("table.col1");
}

// End of File
```

When the telecom bill arrives ...

"We've either got a system error or worse... a hacker?"



Unauthorized access!

A network manager's nightmare. Armed with the right tools, a network manager can quickly get to the bottom of this. Our complete set of network diagnostic tools and

unparalleled customer support are designed to assist you through the toughest network challenges. Supportable PC internetworking technology from designers who've been there.

We're at 1-800-4-NETCOM
(1-800-463-8266).



ADDING RELIABILITY TO UNIX INTERNETWORKING

Phone (416) 856-0238 Fax (416) 856-0242 Internet: netcom2@software.group.com

✧ Request 190 on Reader Service Card ✧

being created, and much time being spent allocating arrays of zero length. To alleviate this situation, we established a special case where a zero-length list was represented by a null implementation pointer (see Figure 8.). Copying and assignment are slightly longer but the default constructor is simplified. Only after using the profiler did we become aware of this special case optimization.

Overloading *new* and *delete*

Another strategy for addressing memory allocation bottlenecks is to overload the *new* and *delete* operators for a class. Using a special purpose memory allocation algorithm rather than the general pur-

using a different heap for different allocation sizes you can reduce the amount of time needed to search for a chunk of free memory, and reduce fragmentation as well.

An example of this is presented in Listings 1 - 3, which implement a memory-pool class and a linked list class that uses the memory pool. The memory-pool class is a heap for a specific size allocation. It is used by associating one instance of the memory pool class with any class (via a static data member) and overloading the *new* and *delete* operators to use the pool. Since the specific *new* operator is only used for that class member, the size is always the same.

The heap works by allocating 32 objects at a time, and maintaining a bit map

and the other for chunks that are completely used up. Therefore the allocation logic has to search only one chunk for a free spot. (I have to thank my brother Mark for the bit searching method, without which this method was actually slower than the built-in *malloc*.) This special version of *new* could also be rewritten in assembly language for even greater speed.

For classes that hold limited resources (such as our *BMOJoin* which controls one or more network sockets), a good strategy is to delay creation until the last possible moment. We used code similar to that in Figure 4b inside the *BMOIterator*, calling *GetJoinPointer* rather than accessing the *BMOJoin* pointer itself. In this way we can create *BMOIterators* as we put up a window, but the *BMOJoin* isn't created until a

Figure 8 Using a special case for an empty list

```
class AttributeList {
private:
    struct AttributeListImp {
        int *list;
        unsigned count;
        AttributeListImp(int *
            data, unsigned size)
        {
            list = new int[size];
            memcpy(list,data,
                sizeof(int)*size);
            count = 1;
        }
    } *ptr;
public:
    AttributeList()
    {
        // Empty list special case:
        ptr = NULL;
    }
    AttributeList(int *data,
        unsigned size)
    {
        ptr = new
            AttributeListImp(data,
                size);
    }
    AttributeList(const
        AttributeList &a)
    {
        // Now we have to check
        // for null:
        ptr = a.ptr;
        if(ptr) ptr->count++;
    }
    ...
};

// End of File
```

Figure 9 Delaying creation of an object by restricting access

```
class BMOJoin;

class BMOIteratorImp {
public:
    BMOIteratorImp()
    {
        itsJoinPtr = NULL;
    }
    ~BMOIteratorImp()
    {
        delete itsJoinPtr;
    }
    int Search(char *str)
    {
        return
            GetJoinPointer()->Search(str);
    }
    void Disconnect()
    {
        delete itsJoinPtr;
        itsJoinPtr = NULL;
    }
private:
    // this is actually split apart
    // (see Figure 4.)
    BMOJoin *GetJoinPointer()
    {
        if(itsJoinPtr)
            return itsJoinPtr;
        itsJoinPtr = new
            BMOJoin(itsStoredParameters);
        if(itsJoinPtr == NULL)
            // for now throw is just an inline
            // for exit()
            throw(ErrNoMem);
        return itsJoinPtr;
    }
};

// End of File
```

Listing 2 continued

```
void *MemoryPoolLink::malloc(
    size_t size)
{
    static char lookup[] = {
        0, 1, 0, 2,
        0, 1, 0, 3,
        0, 1, 0, 2,
        0, 1, 0, 4,
    };
    int shift = 0;
    unsigned long b = bits;
    if((b&0xFFFF) == 0xFFFF)
    {
        shift = 16;
        b >>= 16;
    }
    if((b&0xFF) == 0xFF)
    {
        shift += 8;
        b >>= 8;
    }
    if((b&0xF) == 0xF)
    {
        shift += 4;
        b >>= 4;
    }
    shift += lookup[b&0xF];
    bits |= (1l << shift);
    return data +
        (shift * size);
}

void MemoryPoolLink::free(
    void *ptr,
    size_t size)
{
    bits &=
        ~(1l <<
            ((char *)ptr-data)/size);
}
```

Listing 2 *continued*

```

MemoryPool::MemoryPool(
    size_t _size)
: size(_size),
  freeHead(0, NULL),
  usedHead(0, NULL)
{
}

void *MemoryPool::add()
{
    MemoryPoolLink *temp = new
        MemoryPoolLink(size,
        freeHead.next);
    if(temp == NULL)
        return NULL;
    freeHead.next = temp;
    return
        freeHead.next->
        malloc(size);
}

void *MemoryPool::malloc()
{
    if(freeHead.next)
    {
        void *ret =
            freeHead.next->
            malloc(size);
        if(freeHead.next->bits ==
            0xFFFFFFFF)
        {
            MemoryPoolLink *temp =
                freeHead.next;
            freeHead.next =
                temp->next;
            temp->next =
                usedHead.next;
            usedHead.next = temp;
        }
        return ret;
    }
    return add();
}

void MemoryPool::free(
    void *ptr)
{
    MemoryPoolLink *temp =
        freeHead.next;
    MemoryPoolLink *prev =
        &freeHead;
    while(temp)
    {
        ptrdiff_t diff =
            temp->data -
            (char *)ptr;
        if(diff > 0 &&
            diff < size*PoolSize)
        {
            temp->free(ptr, size);
            if(temp->bits == 01)
            {
                prev->next =
                    temp->next;
                delete temp;
            }
            return;
        }
        prev = temp;
    }
}

```

Listing 2 *continued*

```

        temp = temp->next;
    }
    temp = usedHead.next;
    prev = &usedHead;
    while(temp)
    {
        ptrdiff_t diff =
            temp->data -
            (char *)ptr;
        if(diff > 0 &&
            diff < size*PoolSize)
        {
            temp->free(ptr, size);
            prev->next =
                temp->next;
            temp->next =
                freeHead.next;
            freeHead.next =
                temp;
            return;
        }
        prev = temp;
        temp = temp->next;
    }

    // End of File

```

Listing 3 *A linked list for the memory pool*

```

#include "mempool.h"
#include <fstream.h>
#include <string.h>

class Link {
    char *data;
    Link *next;
    static MemoryPool pool;
public:
    friend class List;
    friend class Iterator;
    Link(const char *_data,
        Link *_next)
    {
        next = _next;
        data = strcpy(new
            char[strlen(_data)+1],
            _data);
    }
    ~Link()
    {
        delete data;
    }
    void *
        operator new(size_t)
    {
        return pool.malloc();
    }
    void
        operator delete(void *ptr)
    {
        pool.free(ptr);
    }
};

```

Listing 3 *continued*

```

MemoryPool
Link::pool(sizeof(Link));

class List {
    Link head;
public:
    friend class Iterator;
    List ();
    ~List();
    void add(char *data);
};

class Iterator {
    Link *link;
public:
    Iterator(List &_list)
    {
        link = _list.head.next;
    }
    char *Next()
    {
        char *ret = link ?
            link->data : NULL;
        link = link->next;
        return ret;
    }
};

List::List()
: head("", NULL)
{
}

List::~List()
{
    while(head.next)
    {
        Link *temp =
            head.next->next;
        delete head.next;
        head.next = temp;
    }
}

void List::add(char *data)
{
    head.next = new
        Link(data, head.next);
}

main(int, char **argv)
{
    static char buff[1024];
    List list;
    ifstream in(argv[1]);
    while(in.getline(buff,
        sizeof(buff)))
        list.add(buff);
    Iterator it(list);
    char *ptr;
    while((ptr = it.Next()) !=
        NULL)
        cout << ptr << endl;
}

// End of File

```

search is performed. We added a member function called *Disconnect* to the *BMOIterator*, and call it when a window is put into the background to delete the *BMOJoin* pointer and free resources for the foreground window (see Figure 9).

Figure 10 *Going low-level to improve efficiency*

Figure 10a. Results of an inefficient += operator.

```
1.0235 500 void PadWithSpaces(CString &str, int count)
{
5.0346 20000 while(count--)
30.0397 19000 str += " ";
1.4354 500 }
```

Figure 10b. Improved performance with low-level C-strings.

```
1.0233 500 void PadWithSpaces(CString &str, int count)
{
2.4505 500 char *tmp = new char[count+1];
3.2930 500 memset(tmp, ' ', count);
0.9438 500 tmp[count] = '\0';
5.3049 500 str += tmp;
1.9430 500 delete[] tmp;
1.4353 500 }
```

// End of File

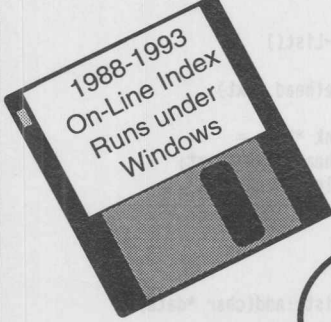
Conclusion

Finally, running the profiler through a particularly sluggish area, we discovered that a lot of time was being spent padding strings with spaces. The problem here was that *operator+=* for the string was not as efficient as we had hoped. By going to a lower level, and accessing C-style strings directly, we managed to speed up the padding process considerably.

I should mention that a good string class would have had a padding function, or at least the ability to create a blank string of count bytes. This low-level access then wouldn't be necessary. Since the string class was part of a commercial library we didn't want to modify or add to it. It's good to know that all the old C tricks are available even if just used at the lowest levels. Credit has to be given here to the profiler for finding this bottleneck.

Many of the optimizations described above are fairly simple because of the separation of interface from implementation. The idea is to be able to grease up a class without having to worry about side effects that might be possible in a less restrictive interface. Other optimizations are possible by dipping into C++'s C heritage as a low-level language.

Of course there is no substitute for a good design effort up front. A temptation in design is to make a lot of "friendly" classes, which access each other's internals. But what you wind up with is "spaghetti objects." By maintaining integrity between classes, you can twiddle with the internal bits to your heart's content without having to worry about unforeseen side effects. □



On-Line Index Updated for 1993 !

Access

The C Users Journal & Windows/DOS Developer's Journal Information Instantly

Find 6 years of indepth real-world information in seconds.
Unleash the full potential of your **C User's Journal** and
Windows/DOS Developer's Journal library.

Only
\$29.95



- Detailed three-level subject index.
- Each article, letter, and question and answer listed under numerous indexing terms.
- All entries include the title, author, and references to other articles, journal issue, and page.
- Select and print the entries you want.

Special Offer

For Owners of the 1992 Index,
buy the update for only \$9.95!
Order W63DU

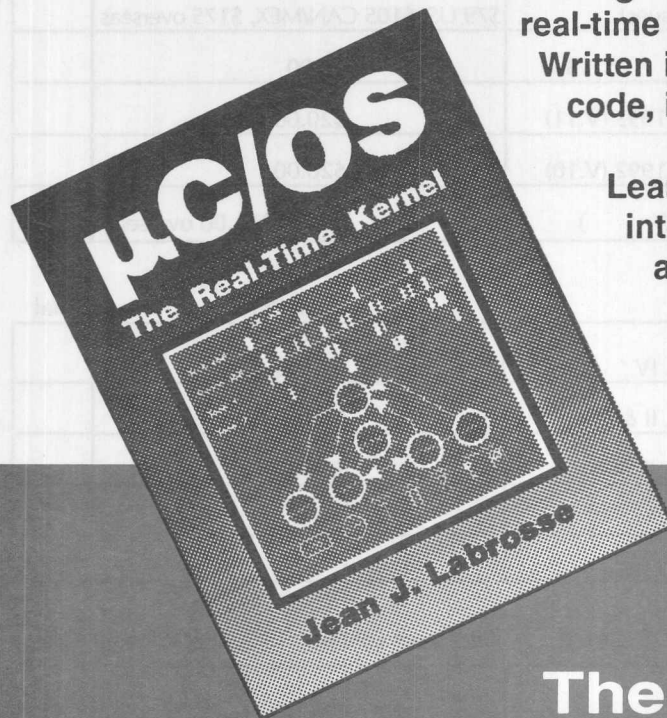
Identify Source Code **INDEXC1** to Order **W63D** Today!

1601 W. 23rd., Suite 200, Lawrence, KS 66046
To Order Call 913-841-1631 FAX 913-841-2624

Push the Limits of Real-time Design!

Investigate the fundamentals of building real-time embedded kernels with μ C/OS. Written in C with minimum assembly code, it is portable and ROM able.



Learn about task priority scheduling, intertask communication, interrupts, and performance benchmarking.

μ C/OS is complete preemptive multitasking operating system that handles 63 tasks.

μ C/OS

The Real Time Kernel

by Jean J. Labrosse

Secrets of Embedded Systems Revealed!

- Performance compares to commercial kernels
- Written in C with assembly code minimized
- Assembly code minimized for easy portability
- Includes System Code & Users Manual



Companion Disk for \$24.95

Order Today!

(order book W60, book & disk W62)

913-841-1631

FAX 913-841-2624

R&D
publications, inc.



The C Users Order Form

	Quantity	Code #	Item	Unit Price	Total
C Users Journal			1-year subscription (12 issues)	\$29.95 US, \$46 CAN/MEX, \$65 overseas	
			2-year subscription (24 issues)	\$56 US, \$76 CAN/MEX, \$123 overseas	
			3-year subscription (36 issues)	\$79 US, \$105 CAN/MEX, \$175 overseas	
			Magazine Code Disk (Mo: Yr:)	\$5.00	
			Magazine Code Disks — 1993 (V.11)	\$20.00	
			Magazine Code Disks — 1992 (V.10)	\$20.00	
			CUJ Back Issue (Mo: Yr:)	\$7.50, \$11.00 overseas	

	Quantity	Code #	Title	Unit Price	Total
Books See below for overseas shipping charges.		CAT4	The CUG Directory - Vol. IV	\$10.00	
		CAT23	The CUG Directory - Vol. II & III	\$18.00	
		CAT24	CUG Directory Volumes I, II, III & IV	\$28.00	

	Quantity	Code #	Disk Format/Size (MS-DOS, etc.)	# disks per volume	Unit Price	Total
Disks See below for overseas shipping charges. NOTE: Please indicate format and size of disks ordered.				() x \$4.00		
				() x \$4.00		
				() x \$4.00		
				() x \$4.00		

Please send more information about:

- ☐ Sys Admin ☐ Windows/DOS
☐ The C Users Journal ☐ The C Users' Group
☐ The C Users Bookstore

All payments must be in US dollars (Mastercard/VISA accepted)
NOTE: Call 913-841-1631 for special shipping!

Name _____
 Company _____
 Address _____
 City _____
 State _____ Zip Code _____
 Country _____
 Daytime Phone _____

☐ I am a current subscriber

OVER-SEAS SHIPPING	BOOKS - Add 45% + \$3.50 s&h	
	DISKS - Add 30% + \$3.50 s&h	
shipping charge in North America (\$3.50)		

TOTAL

☐ MC ☐ Visa Expiration Date _____
 Card # _____
 Signature _____

Mail to: The C Users Journal, 1601 W. 23rd St., Suite 200,
 Lawrence KS 66046-2743, (913) 841-1631
 — For fastest service, FAX (913) 841-2624 —

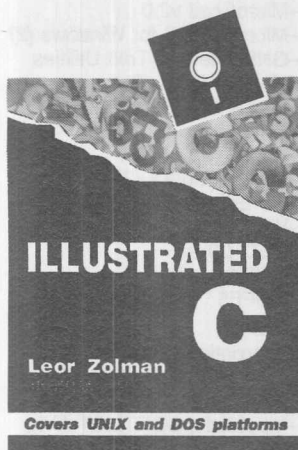


C Users Bookstore

The Resource Center for C Programming Books

Hard-to-find titles and titles you won't find anywhere else

C++ • UNIX • ANSI/Standard C • Turbo C • Windows • X Window



Illustrated C

By Leor Zolman

Illustrated C explores the construction of several different applications, from start to finish. Through a focus on building useful, practical programs, this book gives the C programmer the "why and how" of application design and development. Each program is exhaustively annotated in a clear, readable style. Illustrated C is a tutorial for the novice to intermediate programmer and a toolbox for all C programmers.

R&D Publications, 1992, 320 pp.
ISBN 0-923667-21-0

W34\$29.95

W36 w/ disk\$39.95



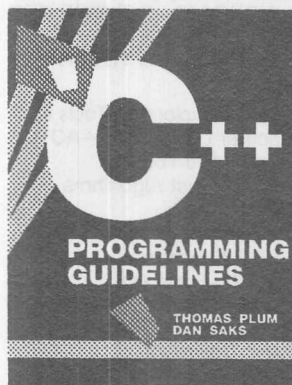
MS-DOS System Programming / 2nd edition

Edited by Robert Ward

This highly technical, focused work is written by working programmers for working programmers and designed to save the reader hours of work. How-to topics include: critical error handling, customizing the DOS boot strap, interrupt-driven I/O, manipulating environmental variables, event timing, writing TSRs and device drivers, and much more. The final chapter is an annotated bibliography of books designed to help the serious programmer develop a personal library of professional MS-DOS literature.

R&D Publications, 1991, 240 pp.
ISBN 0-923667-20-2

Z82\$24.95



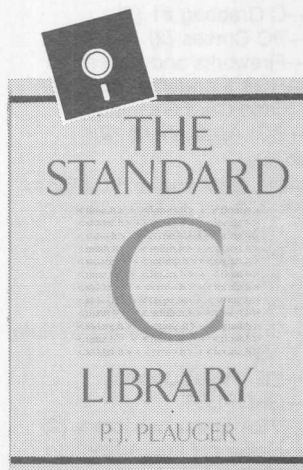
C++ Programming Guidelines

By Thomas Plum and Dan Saks

C++ Programming Guidelines gives professional C++ developers useful information for writing portable and maintainable C++ programs. Written for the professional C and C++ programmer, this book aids the production of efficient, portable programs which can be understood and maintained by other developers. Users learn both lexical layout and more fundamental semantics for data and variables, operators, control statements, and functions. C++ Programming Guidelines is arranged in a "manual-page" format for easy reference.

Plum-Hall, 1992, 265 pp.
ISBN 0-911537-10-4

W33\$34.95



The Standard C Library

By P.J. Plauger

The Standard C Library shows you how to use all of the library functions mandated by the ANSI and ISO Standards for the programming language C. To help you understand how to use the library, this book also shows you how to implement it. Approximately 9,000 lines of tested, working code that is highly portable across diverse computer architectures is included.

Prentice Hall, 1991, 498 pp.
ISBN 0-13-131509-9

Z81\$28.00

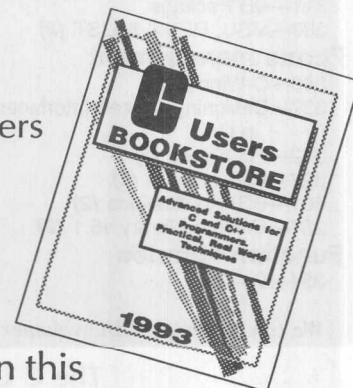
DK3 disk\$99.95

Use the order form on the opposite page to order these titles or any of 150 books in the Bookstore.

FREE 1993 C Users Bookstore Guide

Get your complete listing and description of all the books in the C Users Bookstore. The 16-page reference guide is yours for the asking. Titles are arranged by subject for easy reference and are indexed by title and author.

CALL TODAY—(913) 841-1631. Or, use the Reader Service card in this magazine. Request 153 on the Reader Service Card.



The C Users' Group collects and maintains the public domain C source code and shareware listed below. Distribution fee is \$4.00 per disk. There is a \$3.50 shipping and handling charge per order U.S.; 30% overseas. If there are multiple disks in a volume, that number is indicated in parentheses. Due to updates, the number of disks per volume is subject to change without notice. The CUG Library supports code written for MS-DOS. Be sure to specify size when ordering disk (3½" or 5¼"). Volumes are continually added.

For volumes 101-277 see the CUG Library Directories.

Artificial Intelligence

- 299—MEL and BP
- 297—Small Prolog (2)
- 396—NNUTILS

Assemblers and

Disassemblers

- 398—ASxxxx Cross Assembler - Part 3
- 363—68020—Cross Assembler (2)
- 348—8048—Disassembler/Z80—Assembler
- 346—ASxxxx Cross Assembler - Part 2 (2)
- 338—68000—C compiler and assembler (2)
- 335—Frankenstein Cross Assemblers (4)
- 316—AS8—Cross Assembler
- 303—MC68K Disassembler
- 292—ASxxx Cross Assembler - Part 1 (4)

Communications

- 380—JMODEM
- 314—MNP C Library
- 308—MSU, REMZ & LIST (2)
- 307—ADU & COMX

Compilers & Interpreters

- 359—GNU C/C++ 386—exec and lib. source (12)
- 338—68000—C Compiler and Assembler (2)
- 309—6809—C Compiler for MS-DOS

File management

- 387—C/C++ Lost Algorithms
- 386—Thomson-Davis Editor
- 382—GZIP
- 379—ZOO
- 377—DSR Functions
- 367—GNU File and Text Utilities for MS-DOS (2)
- 358—cbase
- 347—TAVL Tree
- 342—I8255—Interface Library
- 329—UNIX Tools for PC (2)
- 326—SoftC Database Library (3)
- 311—DB Package
- 308—MSU, REMZ & LIST (2)

Forms management

- 340—C-Window
- 337—Designing Screen Interfaces in C
- 332—PC curses (2)
- 327—Panels for C (2)
- 301—BGI Applications (2)
- 281—Unicorn Library v5.1 (2)

Function libraries

- 394—C++SIM

- 395—Input-Edit, SORTLIST AVL, and Typing Tutor
- 393—LL, GIFSave, and Cordic++
- 390—Another C Tools Lib—ACTLIB
- 385—BCC+ Coroutines
- 384—Ghostscript (13)
- 372—Mouse++, String++ and Z++ classes
- 376—OS/2—Tools (4)
- 368—GNUlib for MS-DOS
- 361—Gadgets and Term
- 352—String and Vlist
- 342—I8255—Interface Library
- 339—CTRLCLIB (2)
- 321—Mouse Trap Library
- 309—6809—C Compiler for MS-DOS
- 302—3-D Transforms
- 300—MAT_LIB
- 295—blkio Library

Games

- 344—C Grabbag #1 (2)
- 332—PC Curses (2)
- 323—Fireworks and Adventure
- 305—HGA Mandelbrot Explorer and Card Games (2)
- 301—BGI Applications (2)

Graphics

- 393—LL, GIFSave, and Cordic++
- 389—VGA FontLib, Make Font & DXF viewer
- 383—VGL (2)
- 381—JPEG
- 350—PCX Graphics Library
- 343—C Image Processing System (3)
- 336—EGAPAL/EDIPAL
- 334—GNUPLLOT
- 325—VGA Graphics Library (2)
- 323—Fireworks and Adventure
- 321—Mouse Trap Library
- 320—Convolution Image Process
- 317—Group 3—Image Processing
- 315—FTGRAPH (Fast-Fourier Transform Graphics)
- 305—HGA Mandelbrot Explorer and Card Games (2)
- 302—3-D Transforms
- 301—BGI Applications (2)
- 293—& CUG294—3D Medical Imaging (6)
- 287—GRAD for MSC C (2)
- 286—GRAD for Turbo C (3)

Language Tools & Other Languages

- 391—C/C++ Exploration Tools v2.12
- 376—OS/2—Tools (4)
- 369—Genitor (3)
- 370—GATool (2)
- 364—C-ACROSS (2)
- 356—Serlock for Macintosh (3)

- 355—Sherlock for MS-DOS (3)
- 357—CSTAR (2)
- 345—TLC/TLP
- 344—C Grabbag #1 (2)
- 333—gAWK (2)
- 329—UNIX Tools for PC (2)
- 322—Doctor's Tools
- 319—CPP (C preprocessor) (2)
- 311—Make-Maker
- 310—Little Smalltalk for MS-DOS (2)
- 299—MEL and BP
- 297—Small Prolog (2)
- 296—C to C++ Migrator
- 290—FLEX (2)
- 285—BISON for MS-DOS (3)

Math & Scientific

Applications

- 391—C/C++ Exploration Tools v2.12
- 378—NEWMAT
- 369—Genitor (3)
- 370—GATool (2)
- 344—C Grabbag #1 (2)
- 341—Orbit Propagation
- 315—FTGRAPH (Fast-Fourier Transform)
- 308—MSU, REMZ & LIST (2)
- 306—Thread and Synapsys
- 305—HGA Mandelbrot Explorer and Card Games (2)
- 301—BGI Applications (2)
- 300—MAT_LIB
- 299—MEL and BP

Multitasking

- 387—C/C++ Lost Algorithms
- 385—BCC+ Coroutines
- 362—RMAXTask
- 330—CTask (3)
- 306—Thread and Synapsys

Screen managers

- 340—C-Window
- 337—Designing Screen Interfaces in C
- 328—WTWG (2)
- 327—Panels for C (2)
- 324—WGCONIO
- 321—Mouse Trap Library
- 301—BGI Applications (2)
- 298—PC Curses
- 291—JJB For Quick C and Turbo C Programmers
- 283—FAFNIR (3)

Simulations

- 394—C++SIM
- 349—Simulation Subroutine Set
- 308—MSU, REMZ & LIST (2)
- 306—Thread and Synapsys

Text processing tools

- 399—MINED Editor
- 395—Input-Edit, SORTLIST AVL, and Typing Tutor

- 392—GNU Indent v1.8
- 388—Anthony's Tools
- 386—Thomson-Davis Editor
- 374—MicroSpell v2.0
- 373—MicroEMACS for Windows (2)
- 367—GNU File and Text Utilities for MS-DOS (2)
- 366—MicroEMACS Update (3)
- 365—Elvis (3)
- 360—Uspell (2)
- 344—C Grabbag #1 (2)
- 331—SE Editor
- 324—WGCONIO
- 318—RED (2)
- 313—STEVIE
- 304—ROFF5

Tutorials

- 397—International Obfuscated C Code Contest 1984-1993
- 396—NNUTILS
- 395—Input-Edit, SORTLIST AVL, and Typing Tutor
- 353—C++ Tutor (2)
- 257—& CUG258—C TUTOR FOR TURBO C
- 252—& CUG253—C TUTOR

Utilities

- 391—C/C++ Exploration Tools v2.12
- 390—Another C Tools Lib—ACTLIB
- 388—Anthony's Tools
- 387—C/C++ Lost Algorithms
- 382—GZIP
- 379—ZOO
- 377—DSR Functions
- 376—OS/2—Tools (4)
- 344—C Grabbag #1 (2)
- 339—CTRLCLIB (2)
- 332—P C curses (2)
- 329—UNIX Tools for PC (2)
- 322—Doctor's Tools
- 321—Mouse Trap Library
- 307—ADU and COMX
- 295—blkio Library

Windowing Packages

- 388—Anthony's Tools
- 386—Thomson-Davis Editor
- 375—TextView
- 371—Windos IO v 2.0 (2)
- 361—Gadgets and Term
- 354—CES Mouse Tools Library with JoyStick
- 351—UltraWin (2)
- 340—C-Window
- 337—Designing Screen Interfaces in C
- 332—P C curses (2)
- 328—WTWG (2)
- 327—Panels for C (2)
- 324—WGCONIO
- 298—PC Curses
- 283—FAFNIR (3)

We can supply all CUG volumes 100 through 364 on CD-ROM. Order R01 for \$49.95

The C Users' Group, 1601 W. 23rd St., Suite 200, Lawrence, KS 66046-2700
(913) 841-1631 • FAX (913) 841-2624

Enhancing the UNIX Korn Shell Using Predictor Techniques

Philip Thomas and Shmuel Rotenstreich

Introduction

On most computer systems, users interact with the system through command-language interpreters called shells. These shells accept user input and interpret them as commands for the system. There are three major UNIX shells: Bourne, C and Korn (kshell). Other shells include command.com for MS-DOS and for OS/2.

The Korn Shell, which we consider here, offers sophisticated management of past commands (history). We have enhanced this functionality to include a learning automaton that predicts the next command. This command prediction often allows the user to avoid entering the command sequence in its entirety. In most of the computing environments we considered, this enhanced shell predicted the next command with a high degree of accuracy.

This article describes our shell enhancements and details some of the methods we used to implement them. Although the Korn Shell was our target shell, we have applied these same enhancements to the C Shell. We believe this article will show that our enhancements are applicable to other shells as well.

Attaining Command Cycle Consciousness

In our research we have found that most users exhibit cyclic behavior when interacting with an operating system. For example, to create and execute a program, users may produce the following:

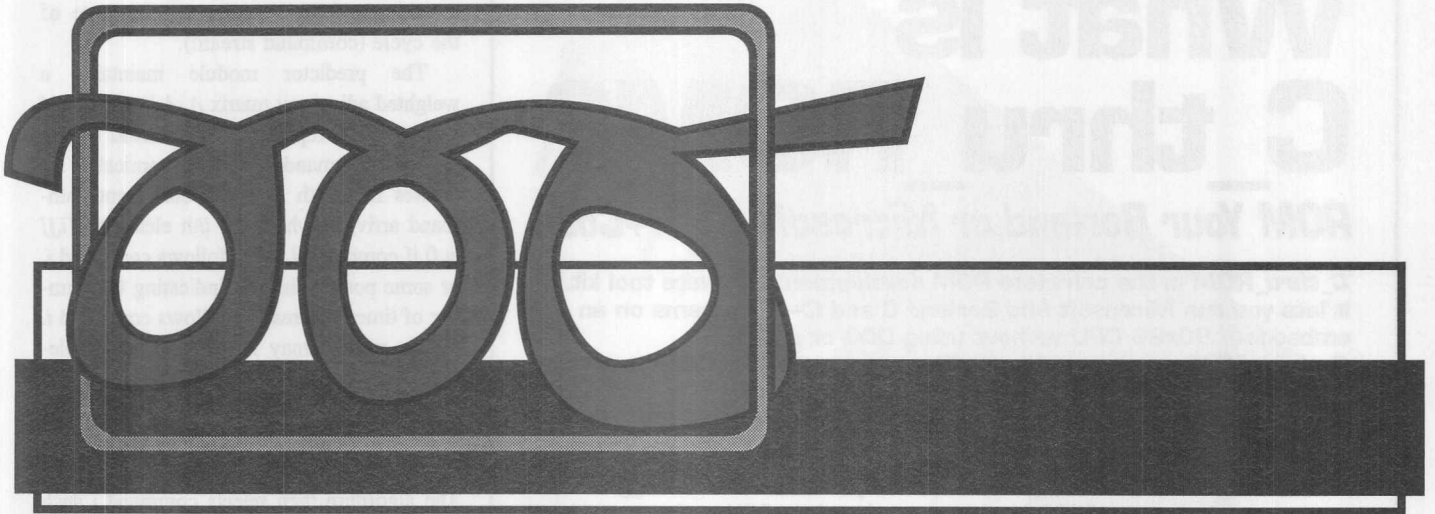
1. Command: Evoke editor — create the source file.
2. Command: Create executable — compile & link the source file.
3. Command: Execute the file created in (2) — examine the results.

The user typically repeats this sequence until he or she has completed and thoroughly debugged the program.

If a shell can recognize such cyclical behavior (we say it has *cycle consciousness*) it can predict the user's next command at any point in the sequence. We incorporate cycle consciousness into the shell by integrating it with the Predictor module outlined in the next section.

The Predictor Module

To incorporate cycle-based knowledge into various operating system facilities, we have developed an abstract data type module called a Predictor (Listing 1). The Predictor is a learning automaton representative of the learning-from-analogy class of learning strategies. In learning-from-analogy strategies, the learner reaches a conclusion about the current case by considering previously processed cases bearing strong similarities to the current case. For the predictor module, the previously processed cases are the sequences of commands already encountered from the command stream.



Philip K. Thomas received the B.S. degree in Physics from the California Polytechnic State University and is currently in the process of defending his Ph.D. dissertation in Computer Science at the George Washington University. Philip also serves as Director of System Architecture at PRC corporation in Mclean, Virginia and can be reached at philip@rsi.prc.com.

Shmuel Rotenstreich received the B.S. degree in Computer Science from the Tel Aviv University and the Ph.D. degree also in Computer Science from the University of California at San Diego. Shmuel is currently Professor of Engineering and Applied Sciences at the George Washington University and can be reached at shmuel@sparko.gwu.edu.

Listing 1 *Definition of the Predictor*

```
enum ReturnValue
{
    Failure,
    Success
};

const DEF_CARDINAL=500; //number of commands a_matrix can hold
const DEF_DELTA_CARDINAL=100; //increment of cardinal_a_matrix
const MAX_SIZE_COMMAND=50; //maximum size of command

class predictor
{
    int num_commands; //current number of unique commands
    int cardinal_a_matrix; //max number of commands a_matrix can hold
    int *a_matrix; //pointer to the adjacency matrix
    char *commands; //list of unique commands
    int last_ordinal; //index of last command

    ReturnValue size(int cardinality); //malloc or realloc to create
    // approp. a_matrix & commands
    int is_unique(char *command); //checks commands[] for command
    // <0 =>No, else=>index of command
    ReturnValue insert(char *command); //add command to commands
    ReturnValue update(int ordinal); //increment a_matrix entry

public:
    predictor()
    {
        num_commands=0;
        cardinal_a_matrix=DEF_CARDINAL;
        size(cardinal_a_matrix);
        last_ordinal=-1;
    }
};
```

The Predictor module incorporates an abstract component as well as several concrete components. The abstract component is the prediction algorithm, which we describe shortly. The concrete components consist of three major classes:

1. **Control Class:** The two instances of this class supply entry points for initializing and terminating the Predictor Module.
2. **Input Class:** This class obtains the next command from external modules.
3. **Predict Class:** This class makes predictions for the Predictor module. *predictor* can make two types of predictions:
 - 1) The next command
 - 2) When a particular command will next occur

By creating a *predictor* class, we encapsulate all cycle-dependent code and data, making it easier to modify. This method of encapsulation also permits the coexistence of multiple predictors. (Environments where multiple command streams exist require multiple predictors.) We describe the *predictor* class in detail later in this article.

The Algorithm

For this discussion we say that a command stream *C* is a sequence of *N* commands composed of *M* unique commands. These commands arrive at the predictor one at a time via the input interface. These commands determine the current state of the cycle (command stream).

The predictor module maintains a weighted adjacency matrix *A*. *A* is an *M* x *M* matrix which represents a command stream *C* (this command stream dynamically increases in length as each subsequent command arrives), where the *i*th element *A*[*i*,*j*] is 0 if command *j* never follows command *i*, or some positive integer indicating the number of times command *j* follows command *i*. (Some readers may recognize this implementation as part of a *semantic network*.)

To predict the next command, the algorithm locates the corresponding row for the latest command *n* in the adjacency matrix *A*. The algorithm then selects command *i* such that command *i* is: $MAX(A[n,i])$ $i=1,2,M$ (index of command *i*) and $A[n,i] > 1$

If such a command exists, the algorithm returns command *i* as the best prediction of the next command to occur. If no such command exists, the algorithm returns indicating that the predictor module cannot predict the next command given its current state.

Free Demo Disk! Call 1-800-221-6630

What is C_thru_ROM?

ROM Your Borland or Microsoft C/C++ Code.

C_thru_ROM is the complete ROM development software tool kit. It lets you run Microsoft and Borland C and C++ programs on an embedded 80x86 CPU without using DOS or a BIOS.

C_thru_ROM saves you money. There are no DOS or BIOS royalties to pay for your embedded systems.

C_thru_ROM is complete! It includes the following and much more:

- Supports Borland's Turbo Debugger.
- Remote Code View style source level debugger.
- ROMable startup code brings CPU up from cold boot.
- ROMable library in source code.
- Flexible 80x86 Locator.

COMPLETE PACKAGE ONLY \$495. 30-DAY MONEY BACK GUARANTEE.

Datalight®

307 N. OLYMPIC, SUITE 201 • ARLINGTON, WA 98223 • (206) 435-8086 • FAX: (206) 435-0253

◆ Request 256 on Reader Service Card ◆

Listing 1 *continued*

```

    }
    predictor(int cardinality) //overload constructor
    {
        num_commands=0;
        cardinal_a_matrix=cardinality;
        size(cardinal_a_matrix);
        last_ordinal=-1;
    }
    ~predictor() //destructor
    {
        cardinal_a_matrix=0;
        free(a_matrix);
        free(commands);
    }

    ReturnValue PutNextCommand(char *);
    ReturnValue GetNextCommand(char *);
    int WhenNextCommand(char *);
};

ReturnValue predictor::PutNextCommand(char *command)
{
    int ordinal;

    if(num_commands+1 >= cardinal_a_matrix) //no space ?
    {
        cardinal_a_matrix+=DEF_DELTA_CARDINAL;
        if(size(cardinal_a_matrix)==Failure)//can not allocate space
            return(Failure);
    }

    if((ordinal==is_unique(command))==-1) //command encountered before?
    {
        if(insert(command) == Failure) //no - add it
            return(Failure);
        ordinal=num_commands; //new ordinal number of command
    }

    return(update(ordinal)); //return appropriate value
}

ReturnValue predictor::GetNextCommand(char *command)
{
    int ordinal,max=0,i;
    int *last_command;

    //note: assumes a_matrix is stored in row major fashion

    last_command=a_matrix+
        last_ordinal*
        cardinal_a_matrix*sizeof(int); //size of a row
    //Note: With certain compilers, sizeof(int) is unnecessary

    for(i=0; i<num_commands; ++i) //do MAX() function
    {
        if(*(last_command+i*sizeof(int))>max)
        {
            max=*(last_command+i*sizeof(int));
            ordinal=i;
        }
    }

    if(max>0)
    {
        strcpy(command,commands*MAX_SIZE_COMMAND);
        return(Success);
    }
    else
    {
        command=NULL;
        return(Failure);
    }
}

```

SLATE

with **Graphics** **SCRIPT & S_PRINT**

Would Text and Graphic Printer support for over 850 printers give you an edge?

By including **SLATE with Graphics**, you can print Text and Graphics on over 850 printers. **Immediately! Painlessly!**

You can use **SLATE** in your product with **no royalties**. It gets you **out of the printer support business**.

Make your product more functional and competitive by using **SLATE's** advanced text features:

- Output to parallel printers, serial printers, DOS files and Novell network printers.
- Support proportional fonts and scalable fonts.
- Set exact print positions.
- Kerning, leading, underlining, and strike through.
- Automatic character set conversion.
- Print lines and shaded areas (laser printers only).

SLATE with Graphics adds advanced graphic printing features:

- Print images from the screen, PCX or TIFF files, or custom image systems.
- Print lines and shaded areas on all printers.
- Scale and Rotate printed image.
- Print grey scale and color images.
- Internix text and graphics.

SLATE is a **C** or **Basic** library of over 170 text printing functions, a **Database of over 850 printers**, and **End User configuration and testing** programs. **SLATE with Graphics** adds over 60 graphic printing functions.

Would a User Configurable Report Writer give you a more competitive product?

SCRIPT is a full featured **Text Formatting** library that can be incorporated into your application. **SCRIPT** uses **SLATE** as its printer driver. **SCRIPT** lets you merge text, data, and **S_PRINT** formatting commands from ASCII files and your application.

Enhance your product by taking advantage of **SCRIPT's** features:

- Allow users to alter document format.
- Set exact positions, center, right adjust, and decimal align.
- Fill paragraphs from unformatted text.
- Add lines, shaded areas, logos, signatures, etc.
- Add commands and macros.

Call or FAX now for a complete catalog and developer's guide for The Symmetry Group's printer support products. **Order SLATE for \$299, SLATE with Graphics for \$448, or SCRIPT for \$199** with our risk free, 30 day return policy.

The
Symmetry
Group

800-346-3938

PO Box 26195
Columbus, OH 43226, USA
614-431-2667 • FAX 614-431-5734

◆ Request 316 on Reader Service Card ◆

Listing 1 *continued*

```

}

int predictor::WhenNextCommand(char *command)
{
    int count=1;
    int old_last_ordinal=last_ordinal; //save state of predictor
    int any_cycle[num_commands];      //check to see if we have cycled
    int i;
    char pcommand[MAX_SIZE_COMMAND];

    //zero
    for(i=0; i<num_commands; ++i)
        any_cycle[i]=0;

    while(GetNextCommand(command)==Success)
    {
        if(!strcmp(pcommand,command)) //found the command
            break;

        i=is_unique(pcommand);        //find index of predicted command
        if(!any_cycle[i]^1)           //test and check
        {
            count=0;                  //we have cycled - failure
            break;
        }

        ++count;
    }
    return(count);
}

/* End of File */

```

The algorithm performs the following steps to predict *when* a particular command will next occur:

1. Save the current state of the predictor module.
2. Set *count*=0.
3. Call the predict-the-next-command interface repeatedly until the return value command *i* is
 - a) Equal to the specified command
 - b) A value indicating that the predictor cannot make a prediction
 - c) A value previously returned by the predict-the-next-command interface (this implies that the predictor assumes the command stream will cycle without an occurrence of the specified command). For each time the predict-the-next-command interface is called, increment *count*.
4. Restore the original state of the predictor module.
5. Return *count* if step 3 terminated because of condition 3.a, otherwise return a value indicating that the predictor cannot predicting when the specified command will next occur.

The Predictor Class

The *predictor* class contains three publicly accessible functions to perform input and output:

PutNextCommand — This function registers commands occurring on the system. We modified the main control loop of the kshell to call this function whenever the kshell assembled a command and registered it to provide “history” functionality. This function returns values corresponding to internal fatal errors and success.

GetNextCommand — This function passes the next predicted command to the system, if it can be predicted. We augmented the kshell source to recognize when the user types ESC-p (in emacs mode) at the prompt, and to call *GetNextCommand*. As a result, the kshell produces either a beep (to indicate the inability to predict the next command) or a command string adjacent to the prompt, which the user may execute by typing a carriage return.

WhenNextCommand — As its name implies, this function predicts when a particular command will occur. We do not use this function in the kshell. We have

Is the code you write today the maintenance nightmare of tomorrow?



It wouldn't be if you used PC-METRIC. Find overly complex classes and functions, assess average variable name length and measure comments. Compare metrics across releases. Easy to use, fast, informative. Text and graphical reports, ad hoc queries, standards enforcement. Used by thousands worldwide. Only \$399 for MS-DOS, \$525 for UNIX.

SET Laboratories, Inc.
PO Box 868 Mulino, OR 97042
503-829-7123 (voice) 503-829-7220 (FAX)
info@setlabs.OR97042.sai.com

included it in the *predictor* module because it is invaluable in other systems that require this kind of prediction. (e.g. in a file caching system — when a file needs “flushing,” information about when a file will next be used is advantageous.) This function returns zero to indicate that the command stream will cycle before it encounters the specified command. This function returns a value of less than zero to signal an internal error.

These last two functions maintain and manipulate the adjacency matrix mentioned previously.

Results

To show how well the predictor works, we have derived the following examples from a command stream in a programming environment. Figure 1 shows the first example. The first column shows the actual command stream. The second column, with the prefix *p:* shows the predicted command stream. The last two columns show the current percentile of successful predictions and the maximum percentile of successful predictions, respectively. The second example (Figure 2) differs from the first only by one command; we have removed this command, *ls*, from the stream in example 2. This command represents an “exception” to the cyclical nature of the command stream. Thus, example 2 shows the effect of filtering out exceptions.

Open Issues:

To keep this article concise several items have not been sufficiently developed. The following are a few points that we believe require further clarification.

How are command parameters handled?

Commands can be classified as either tightly coupled or loosely coupled with their parameters. The degree of coupling is context dependent. For instance, when the predictor is used in a file caching subsystem, predicting the sequence of parameters (e.g. file names) is more important than predicting the sequence of the commands themselves. We say the commands are loosely coupled. In the case of tightly coupled commands, we treat parameters and their commands as one predictor unit, that is, the predictor considers the same command with different parameters

Anatomy of a Command Shell

In most systems, user programs and system programs are executed by the command-language interpreter. In more sophisticated systems such as UNIX, there are no major distinctions between this interpreter and any other program — so users can easily create their own shells.

The shell actually executes a command by completing a *fork*, after which the child process executes an *execve* (load and execute) of the command. The parent process (the shell) does a *wait* and suspends its own execution until the child process finishes executing and performs an *exit*. In multi-tasking systems, such as UNIX,

both the shell and the command it is processing can execute concurrently. Users can initiate concurrent execution by typing a command followed by an ampersand. The shell interprets this symbol as an indication not to perform the *wait*; instead the shell continues with the next step in its command input by prompting the user for the next command.

This capability implies the existence of a fairly sophisticated inter-process communication system. As a bare minimum, a child process needs a method to signal its termination to the parent process (the shell), which is not necessarily polling for this signal. □

THERE IS NO SUCH THING AS A FREE MAKE!

Why pay for a Make utility when one comes free with your compiler? Because the moment you start using it you start paying for it. You pay with your time, effort and frustration in trying to keep your software projects up-to-date.

Opus Make excels where your Make falters. For large projects and interfacing to version control systems, for transparent handling of your Microsoft and Borland makefiles, for PolyMake makefile processing at twice the speed for half the price, for exceptional technical support at no additional charge, Opus Make version 6.0 is the only economical choice.

Your time isn't free. Neither is your sanity. Find out why Andrew Binstock says "*Opus Make is the most compelling reason for not using the make utilities bundled with today's compilers.*"



Opus Make version 6.0 features include:

- Makefile Compatibility:** Processes Intersolv Polymake 4.0™, Microsoft™ NMake and PWB, and Borland Make™ makefiles (even auto dependencies).
- Polymake Emulation:** Emulation of Polymake v4.0!
- Object Library Maintenance:** Maintain libraries without object file clutter - thus saving disk space.
- Version Control Support:** Access source files in One Tree SourceSafe™, Burton Systems TLJB™, Intersolv PVCs™, and MKS RCS™ version control systems.
- Other features:** Multiple-directory support • Memory swapping • Pattern-based inference rules • Multiple targets created from single source • Automatic and in-line response files • Queued shell lines • Conditional, looping and include directives.
- Opus MKMF V3.2 included:** MKMF is our makefile and dependency generator. It quickly builds and maintains your makefiles with a minimum of input. It understands C-preprocessor directives and resource compiler files. If you hate building makefiles by hand this is the tool for you!

OPUS Software, Inc. • 1032 Irving Street, Suite 439 • San Francisco, CA 94122 • USA
Phone: (415)-664-7901 • Fax: (415)-664-5624
For More Information Call 1-800-248-OPUS ext 31

Both DOS
and OS/2
for only
\$165.00

◆ Request 159 on Reader Service Card ◆

command streams

Example 1:

=====

Original Command Stream Predicted Command Stream

o:cc -o timer timer.c	p:	00%	00%
o:timer	p:	00%	00%
o:ls -l timer	p:	00%	00%
Removed from example 2			
o:vi timer.c	p:	00%	00%
o:cc -o timer timer.c	p:cc -o timer timer.c	*16%	16%
o:timer	p:timer	*28%	28%
o:vi timer.c	p:ls -l timer	25%	28%
o:vi tstin	p:cc -o timer timer.c	22%	28%
o:cc -o timer timer.c	p:	20%	28%
o:vi tstin	p:timer	18%	28%
o:vi timer.c	p:cc -o timer timer.c	16%	28%
o:cc -o timer timer.c	p:cc -o timer timer.c	*23%	28%
o:timer	p:timer	*28%	28%
o:vi timer.c	p:vi timer.c	*33%	33%
o:cc -o timer timer.c	p:cc -o timer timer.c	*37%	37%
o:timer	p:timer	*41%	41%
o:vi timer.c	p:vi timer.c	*44%	44%
o:cc -o timer timer.c	p:cc -o timer timer.c	*47%	47%
o:timer	p:timer	*50%	50%
o:vi timer.c	p:vi timer.c	*52%	52%
o:cc -o timer timer.c	p:cc -o timer timer.c	*54%	54%
o:timer	p:timer	*56%	56%
o:vi timer.c	p:vi timer.c	*58%	58%
o:cc -o timer timer.c	p:cc -o timer timer.c	*60%	60%
o:timer	p:timer	*61%	61%
o:vi timer.c	p:vi timer.c	*62%	62%
o:cc -o timer timer.c	p:cc -o timer timer.c	*64%	64%

either parse the parameters out and use them as distinct predictor units, or, as is usually more beneficial, we still treat command-parameter sets as tightly coupled and treat them as single predictor units.

Are command cycles common?

Yes. When we remove a lot of the "noise" from command streams, we find that most of the commands are members of command cycles. The "noise," we have discovered, comes in many forms. Many command streams are sprinkled with commands we isolate as "exceptions." These exceptions would be commands like ls, dir, who etc. Users tend to use commands like these with enough randomness that it is easy to filter these commands out of otherwise perfectly healthy command streams. Another form of noise is the partial cycle. The partial cycle is best illustrated using the example session. In the edit-compile-execute cycle mentioned above, evoking the compiler sometimes results in errors that require the user to re-visit the editor without completing the cycle. Partial cycles are much harder to deal with than exceptions and, consequently, our success rate with these tends to be lower.

GUI Environments

We have also begun investigations into incorporating cycle consciousness into graphical user interface (GUI) environments.

Virtuoso™

Giving you the time to design in real-time.

Virtuoso delivers ! Ported from 8bit microcontrollers to parallel processing systems with thousands of 32bit DSPs. Ease of use, superfast performance, high reliability and value for money. Includes source code, 12 months support and maintenance. Ask for full details and availability.

Virtuoso Nano

The fastest and smallest multi-tasking kernel for DSPs. Process switching and services in < 1 microsecond. Stdio. Network wide messages.

Virtuoso Micro

A high level but small preemptive real-time kernel with fully distributed semantics for transparent parallel programming. An industries' first with built-in cross development capability in C on any target ! With task level debugger, tracing monitor, stdio, etc. Free license for use under DOS.

Virtuoso Classico

A seamless integration of Virtuoso Nano and Micro. A must for parallel processing systems. The real-time O.S. that really works for DSPs.

Virtuoso Modulo 0, I, II, III, IV, V, VI

A wide range of optimized libraries and support tools for DSP (> 1000 functions, stdio, PC graphics, plotting, heap allocation, loader, etc.)

Worldwide distributed, our open distribution policy welcomes new representatives and Value Added Resellers. Join the Virtuoso concept now !

Intelligent Systems International. Tel. (+32).16.62.15.85.
Fax (+32).16.62.15.84. e-mail : virtuoso@bix.com

◆ Request 206 on Reader Service Card ◆

Patching the Korn Shell

The kshell source we started with was PD KornShell written by Eric Gisin <egisin@math.UWaterloo.EDU>. PD Korn Shell installs on 4.2+ BSD System V, and POSIX-compatible systems. PD KornShell assumes you have Standard C (ANSI) and POSIX header files and functions. The PD KornShell source is available on several Internet locations including:

ftp.uu.net (192.48.96.9)

/usenet/comp.sources.amiga/volume91/shells and

softu1.ncu.edu.tw (140.115.19.11)

/pub5/tarz

We assembled the kshell on an HP-UX machine using HP's C++ compiler. On the HP-UX system, which is a hybrid UNIX system (System V & BSD), only a negligible amount of source modification was needed to get the shell up and running. Since the source was in "public-domain UNIX-type" C, we decided to leave the original parts in C and incorporate our enhancements in a C++ class. To facilitate the intermingling of C and C++, we instructed the C++ compiler to suppress name mangling (See "Using C/C++ with Clipper" by Mark W. Schumann in the December 1993 issue of CUJ for a thorough treatment of name mangling in C++). □

The essential problem here is defining a command primitive. In a command-oriented interface a command primitive is well defined, but in a modern GUI a command primitive is more loosely defined and may incorporate several user actions, such as mouse moves and button clicks. User activity is hard to partition into distinct commands.

Conclusion

As in any article of this nature, we had to summarize a large body of work both theoretical and experimental to present here. The algorithm presented here is a first-degree learning automaton. We have added several filters and general enhancements to this automaton resulting in several different predictors that, depending on the situation, are selected and evoked at run time (late binding — virtual functions). Our work leveraged the *predictor* across

Figure 1 continued

o:timer	p:timer	*65%	65%
o:vi timer.c	p:vi timer.c	*66%	66%
o:cc -o timer timer.c	p:cc -o timer timer.c	*67%	67%
o:timer	p:timer	*68%	68%
o:vi timer.c	p:vi timer.c	*69%	69%
o:cc -o timer timer.c	p:cc -o timer timer.c	*70%	70%
o:timer	p:timer	*71%	71%
o:vi timer.c	p:vi timer.c	*72%	72%
o:cc -o timer timer.c	p:cc -o timer timer.c	*72%	72%
o:timer	p:timer	*73%	73%
o:vi timer.c	p:vi timer.c	*74%	74%
o:cc -o timer timer.c	p:cc -o timer timer.c	*75%	75%
o:timer	p:timer	*75%	75%
o:vi timer.c	p:vi timer.c	*76%	76%
o:cc -o timer timer.c	p:cc -o timer timer.c	*76%	76%
o:timer	p:timer	*77%	77%
o:vi timer.c	p:vi timer.c	*77%	77%
o:cc -o timer timer.c	p:cc -o timer timer.c	*78%	78%
o:timer	p:timer	*78%	78%
o:vi timer.c	p:vi timer.c	*79%	79%
o:timer	p:cc -o timer timer.c	77%	79%
o:vi timer.c	p:vi timer.c	*78%	79%
o:timer	p:cc -o timer timer.c	76%	79%
o:vi timer.c	p:vi timer.c	*76%	79%
o:cc -o timer timer.c	p:cc -o timer timer.c	*77%	79%
o:timer	p:timer	*77%	79%
o:vi timer.c	p:vi timer.c	*78%	79%
o:timer	p:cc -o timer timer.c	76%	79%
o:mail	p:vi timer.c	75%	79%
o:remsh rigel	p:	74%	79%
o:vi timer.c	p:	72%	79%
o:cc -o timer timer.c	p:cc -o timer timer.c	*73%	79%
o:timer	p:timer	*73%	79%
o:vi timer.c	p:vi timer.c	*74%	79%
o:cc -o timer timer.c	p:cc -o timer timer.c	*74%	79%
o:vi timer.c	p:timer	73%	79%
o:cc -o timer timer.c	p:cc -o timer timer.c	*73%	79%
o:more \$inc/setjmp.h	p:timer	72%	79%
o:vi timer.c	p:	71%	79%
o:cc -o timer timer.c	p:cc -o timer timer.c	*72%	79%
o:vi timer.c	p:timer	71%	79%
o:cc -o timer timer.c	p:cc -o timer timer.c	*71%	79%
o:vi timer.c	p:timer	70%	79%
o:cc -o timer timer.c	p:cc -o timer timer.c	*70%	79%
o:timer	p:timer	*71%	79%
o:vi timer.c	p:vi timer.c	*71%	79%
o:cc -o timer timer.c	p:cc -o timer timer.c	*72%	79%
o:timer	p:timer	*72%	79%
o:vi timer.c	p:vi timer.c	*72%	79%
o:cc -o timer timer.c	p:cc -o timer timer.c	*73%	79%
o:timer	p:timer	*73%	79%

C MemSL™/MemSL++™ C++ Memory Structures Library New Version Collection Classes

Single and Double Linked Lists - cut, copy, insert and append entire list selections. Plus: sorted lists, circular lists, Add, Insert, Append, Delete, Search, Next, Previous, and more.

Hash Tables - linked list or binary tree collision resolution and definable hashing functions.

Multi-dimensional Dynamic Arrays - allocate single or multiple dimension dynamic arrays of any type. Quick sort, linear search and binary search can be used on the arrays.

Binary and AVL Balanced/Threaded Trees - includes several routines for easily managing binary trees. The AVL balanced and threaded tree has all the functionality of a disk-based B-Tree, including: Add, Delete, GetEqual, GetEqGr, GetGreater, GetEqLs, GetLess, GetFirst, GetLast, GetNext, GetPrevious, and more.

Stacks, Queues, Dequeues, Priority Queues, Sets, Bags and Graphs - allow the choice of a linked list of items, or an array of items for a tighter fit.

Definable Memory Handlers - allow user-defined memory allocation routines for third-party memory managers. You can access all the memory you want!

C/C++ Exception Handling - all MemSL functions are designed with plenty of exception handling. Exceptions can be handled for out-of-memory conditions, out-of-range conditions and improper-use conditions. Exception handlers can also be used for warning conditions, and more.

C++ Templates - includes plenty of inline functions and the choice between template and non-template functions. The MemSL can also be compiled to inherit from an object class.

And Much, Much More - the functions and functionality listed here are only a partial list of the MemSL. The MemSL was designed to be portable and very simple to use. No knowledge of data structures is necessary when using the MemSL.

Coming Soon: DiskSL™ and the DiskSL++™.

Support for K&R C, ANSI C and C++
for most compilers and operating systems.

Royalty Free. Source Code Included.

Order On-line: \$99.99

For Demo and Trial Version or to Order On-line

(602) 780-9692

Settings: N, 8, 1

Includes: HP, PS or ASCII Manual.

Shipment: \$139.99*

For Information, FAX on Demand, or to Order

(602) 561-8788

Or FAX to : (602) 561-8106

*Shipment includes Printed Manual,
Diskette and S&H within U.S.

Windbase Software Inc.

P.O. Box 10115

Glendale, Arizona 85318-0115

 **Windbase™**
Software Inc.
Software Development Libraries and Tools

✧ Request 178 on Reader Service Card ✧

several operating-system elements including memory management, schedulings and disk allocation. Overall, we think our assertions for incorporating cycle consciousness have been well founded. We have had favorable results in the areas we have studied. □

References:

Peterson, James L.; Silberschatz, Abraham, *Operating System Concepts*, Addison-Wesley Publishing Co, 1985.

Tanimoto, Steven L., *The Elements of Artificial Intelligence*, Computer Science Press, 1990.

Thomas, Philip K., Rotenstreich, Shmuel, "Command Cycles," Ph.D. Dissertation, George Washington University, 1993.

Figure 2 The sample session with an exception removed

Example 2: (Without ls)

=====

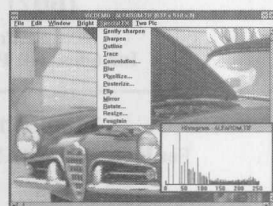
Original Command Stream Predicted Command Stream

Original Command Stream	Predicted Command Stream		
o:vi timer.c	p:	00%	00%
o:cc -o timer timer.c	p:	00%	00%
o:timer	p:	00%	00%
o:vi timer.c	p:	00%	00%
o:cc -o timer timer.c	p:cc -o timer timer.c	*20%	20%
o:timer	p:timer	*33%	33%
o:vi timer.c	p:vi timer.c	*42%	42%
o:vi tstin	p:cc -o timer timer.c	37%	42%

Victor Image Processing Library

Create Powerful Image Applications
for BMP, TIFF, PCX, GIF, TGA, and JPEG Images

- ▶ Load and save BMP/TIFF/PCX/GIF/TGA/JPEG
- ▶ Powerful grayscale and color image processing: brightness, contrast, sharpen, outline, equalize, matrix convolution, rotate, resize, and more
- ▶ Color reduction for fast and accurate display of 24-bit images
- ▶ Support for EGA/VGA/SVGA, 32K- and 16 million-color displays
- ▶ Scan b/w, grayscale, and color images with ScanJet scanners
- ▶ Print halftones, diffusion scatters, and color pictures
- ▶ Convert images between 1-, 8-, and 24-bit formats
- ▶ Convert color to grayscale
- ▶ Includes a complete image processing application with C source



Your Windows application can load and save BMP, TIFF, PCX, GIF, TGA, and JPEG files, control scanner and printer, and have powerful image processing and color reduction for the very best image display.

Victor Image Processing Library for Windows (DLL), \$295

Victor Image Processing Library for DOS, supports Microsoft and Borland C/C++ compilers, \$195

Call or fax to order
314-962-7833

Catenary Systems
470 Bellevue St Louis MO 63119
voice/fax 314-962-7833

Victor Image Processing Library for Windows or for DOS. No royalties. Source code available. visa/mc/cod.

◆ Request 253 on Reader Service Card ◆

Figure 2 continued

o:cc -o timer timer.c	p:	33%	42%
o:vi tstin	p:timer	30%	42%
o:vi timer.c	p:cc -o timer timer.c	27%	42%
o:cc -o timer timer.c	p:cc -o timer timer.c	*33%	42%
o:timer	p:timer	*38%	42%
o:vi timer.c	p:vi timer.c	*42%	42%
o:cc -o timer timer.c	p:cc -o timer timer.c	*46%	46%
o:timer	p:timer	*50%	50%
o:vi timer.c	p:vi timer.c	*52%	52%
o:cc -o timer timer.c	p:cc -o timer timer.c	*55%	55%
o:timer	p:timer	*57%	57%
o:vi timer.c	p:vi timer.c	*60%	60%
o:cc -o timer timer.c	p:cc -o timer timer.c	*61%	61%
o:timer	p:timer	*63%	63%
o:vi timer.c	p:vi timer.c	*65%	65%
o:cc -o timer timer.c	p:cc -o timer timer.c	*66%	66%
o:timer	p:timer	*68%	68%
o:vi timer.c	p:vi timer.c	*69%	69%
o:cc -o timer timer.c	p:cc -o timer timer.c	*70%	70%
o:timer	p:timer	*71%	71%
o:vi timer.c	p:vi timer.c	*72%	72%
o:cc -o timer timer.c	p:cc -o timer timer.c	*73%	73%
o:timer	p:timer	*74%	74%
o:vi timer.c	p:vi timer.c	*75%	75%
o:cc -o timer timer.c	p:cc -o timer timer.c	*75%	75%
o:timer	p:timer	*76%	76%
o:vi timer.c	p:vi timer.c	*77%	77%
o:cc -o timer timer.c	p:cc -o timer timer.c	*77%	77%
o:timer	p:timer	*78%	78%
o:vi timer.c	p:vi timer.c	*78%	78%
o:cc -o timer timer.c	p:cc -o timer timer.c	*79%	79%
o:timer	p:timer	*80%	80%
o:vi timer.c	p:vi timer.c	*80%	80%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	80%
o:timer	p:timer	*81%	81%
o:vi timer.c	p:vi timer.c	*81%	81%
o:cc -o timer timer.c	p:cc -o timer timer.c	*82%	82%
o:timer	p:timer	*82%	82%
o:vi timer.c	p:vi timer.c	*82%	82%
o:timer	p:cc -o timer timer.c	81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82%
o:vi timer.c	p:vi timer.c	*80%	82%
o:cc -o timer timer.c	p:cc -o timer timer.c	*80%	82%
o:timer	p:timer	*81%	82%
o:vi timer.c	p:vi timer.c	*81%	82%
o:timer	p:cc -o timer timer.c	80%	82

The Return Types of Virtual Functions

WG21+X3J16, the joint ISO ANSI C++ technical committee, is now in its fifth year of work on a standard definition for the C++ programming language and its accompanying library. Over the years, the committee has added more than a dozen new features to the language. I described several of them:

- templates
- exception handling
- new keywords and digraphs to support European translation environments
- `wchar_t` as a keyword
- function and operator overloading for enumeration types
- `operator new[]` and `operator delete[]` for array allocation

in "Recent Language Extensions to C++", *CUJ*, June 1993, and described one other:

- forward declaration of nested classes in "Nested Classes", *CUJ*, July 1993. This month, I'll explain another extension:
- relaxed restrictions on the return types for virtual functions

This feature enhances the language's support for object-oriented programming. In particular, it extends the set of valid conversions within a type hierarchy that you can write without casting. I assume you are familiar with virtual function in C++, which I described in my last three columns (see "Virtual Functions," "How Virtual Functions Work," and "Overloading and Overriding" in *CUJ*, December, 1993 through February, 1994).

What the ARM Says

According to the *The Annotated C++ Reference Manual* (ARM) (Ellis and Stroustrup [1990]), a member function de-

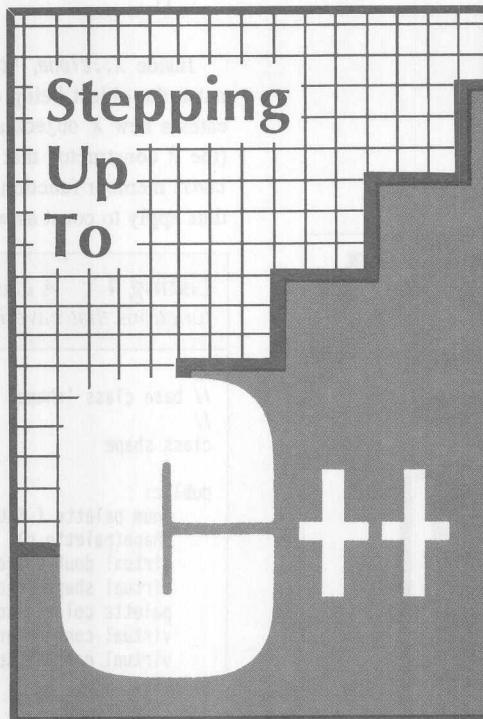
clared in a derived class *D* overrides a virtual function in its base class *B* only if that function in *D* has the same name and signature (sequence of parameter types) as the function it overrides. Clearly, if the name of a function, say *f*, declared in *D* differs from the name of every function declared in *B*, then *f* doesn't override anything. *f*'s declaration adds a new name to the scope of *D*. If *D* declares a function with the same name, again say *f*, as one or

more functions inherited from *B*, but *D*'s *f* has a signature that differs from the signatures of every *f* function in *B*, then again *D*'s *f* doesn't override anything. In this case, *D*'s *f* hides all of *B*'s *f* functions while in the scope of *D*. This is not an error, but as I explained last month, the resulting behavior might surprise you. Consequently, many C++ compilers issue a warning when this sort of hiding occurs.

What about differing return types? According to Section 10.2 of the ARM: "It is an error for a derived class function to differ from a base class virtual function in the return type only." For example, in

```
class B
{
public:
    virtual int vf(int);
    ...
};

class D : public B
{
    ...
    void vf(int); // error
    ...
};
```



Dan Saks is the president of Saks & Associates, which offers consulting and training in C++ and C. He is secretary of the ANSI and ISO C++ committees. Dan is coauthor of C++ Programming Guidelines, and codeveloper of the Plum Hall Validation Suite for C++ (both with Thomas Plum). You can reach him at 393 Leander Dr., Springfield OH, 45504-4906, by phone at (513)324-3601, or electronically at dsaks@wittenberg.edu.

the declaration of *D::vf* is an error because it has the same name and signature as a function declared in *B*, but *B::vf* and *D::vf* have different return types.

Consider the consequences of allowing different return types in this situation. A function *g* that accepts a formal parameter of type *B ** or *B &* can apply *vf* to that *B* object, as in

```
void g(B *bp)
{
    ....
    if (bp->vf(0) > 1)
        ...
}
```

When compiling this function, the compiler considers only the definition for class *B*. None of the classes derived from *B* need to be declared when compiling *g*. Based on the static type of *B::vf*, the call *bp->vf(0)* in *g* should be just fine; it returns an *int*, as the *if* statement in *g* apparently expects.

Since *D* is publicly derived from *B*, you can pass a *D ** to *g*, as in

```
D d;
...
g(&d);
```

But now if *D::vf* has a void return, how can the call *bp->vf(0)* possibly return an *int*? It can't, which is why the ARM insists that an overriding virtual function must have the same return type as the function it overrides.

Cloning Objects

Some members of the standards committee suggested that this rule is more restrictive than it needs to be. There are, in fact, legitimate circumstances where the return types of the overridden and overriding functions need not be absolutely identical. Clone functions are one such circumstance.

Some applications need to be able to clone an object, that is, create an object that's an exact copy of another object. Typically, you implement a class *X* with a cloning function as something like

```
class X
{
public:
    X *clone() const
    { return new X(*this); }
    ...
};
```

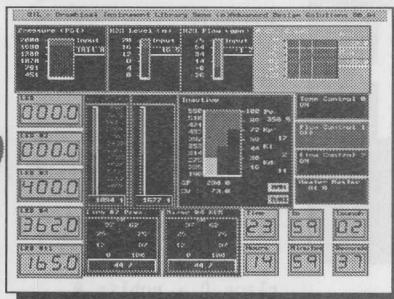
Inside *X::clone*, **this* is an expression of type *X* that designates the object being cloned. The expression *new X(*this)* allocates a new *X* object and initializes it using *X*'s copy constructor (the *X* constructor that takes an argument of type *X*). *clone* is a *const* member function because it does not alter **this*, and should thus apply to *const* as well as non-*const* objects.

GIL 2.6

ONLY \$299

Graphical Instrument Library

The universal library for building fast graphical displays for data acquisition and control.



Royalty Free!

Rich set of instruments includes: Dial gauges, bar gauges, thermometers, seven-segment displays, strip-charts, annunciators, alarms, signal conditioning, 100 timers, PID control and more!

All instruments are fully scalable, 90 degree rotatable, and optimized for maximum throughput, with minimum code size

One version supports Turbo C, Microsoft C, QuickC, QuickBASIC, PDS7.1, & Visual Basic for DOS!

For a free demo, call

(404) 352-4788

ADS Advanced Design Solutions

1920 Moores Mill Rd., Atlanta, GA 30318

* \$249 is a special introductory offer. After August 31, 1991 the retail price will be \$495 U.S.

Listing 1 A class hierarchy with virtual cloning functions that have identical return type

```
//
// base class 'shape'
//
class shape
{
public:
    enum palette { BLUE, GREEN, RED };
    shape(palette c);
    virtual double area() const = 0;
    virtual shape *clone() const = 0;
    palette color() const;
    virtual const char *name() const = 0;
    virtual ostream &put(ostream &os) const;

private:
    palette _color;
    static const char *color_image[RED - BLUE + 1];
};

...

//
// class 'circle' derived from 'shape'
//
class circle : public shape
{
public:
    circle(palette c, double r);
    double area() const;
    shape *clone() const;
    const char *name() const;
    ostream &put(ostream &os) const;
```

◆ Request 242 on Reader Service Card ◆

new X returns an *X **. Although you could write a clone function that returns an *X* or an *X &*, I suggest returning *X ** to emphasize that *clone* returns a dynamically-allocated object that should be deleted eventually. In general, for any class *X*, the return type of *X::clone* should be *X **. For example, in a library of geometric shapes, *circle::clone* should return a *circle ** and *rectangle::clone* should return a *rectangle **.

When used in a class that's the root of a polymorphic hierarchy, clone functions should be *virtual*, and often *pure virtual*. This lets you clone an object without knowing its exact type. For example, Listing 1 shows a polymorphic hierarchy of shapes similar to the one I presented three months ago (see "Virtual Functions", *CUJ*, December 1993). Classes *circle*, *rectangle*, and *triangle* are all derived from base class *shape*. *shape* declares a *pure virtual* clone function as:

```
virtual shape *clone() const = 0;
```

Each of the derived classes overrides the clone function with an impure definition. For instance, class *circle* declares

```
virtual shape *clone() const;
```

in the class definition, and later defines

```
shape *circle::clone() const
{
    return new circle(*this);
}
```

Not long ago I said that, in general, a clone function for a class *X* should have a return type of *X **. But here the return type of *circle::clone* is *shape **, not *circle **. According to ARM, *circle::clone* must return a *shape ** because that is the return type of the function it overrides. Nonetheless, this clone function is still quite useful.

circle::clone returns the result of *new circle*, which is indeed a *circle **. Since *circle* is publicly derived from *shape*, a *circle* is a *shape*, so the *circle ** in the return expression converts safely and quietly to the return type *shape **. A similar conversion occurs in the clone functions for *rectangle* and *triangle* shown in Listing 1. An application can clone an arbitrary shape with code such as:

```
shape *s;
...
shape *cs = s->clone();
```

which leaves *cs* pointing to an object that has the same dynamic type and value as **s*.

Listing 2 shows a more elaborate example dealing with collections of shapes

implemented as arrays of pointers. The *clone_all* function replicates an entire collection of shapes. First, it allocates a new array to hold the pointers to the shape clones. Then, for each shape in the original collection, it clones that shape (using its virtual clone function) and places the pointer to the copy into the new array. As is always the case with polymorphic objects, it doesn't matter to *clone_all* how many different shapes there are; each shape knows how to clone itself.

Unnecessary Downcasting?

The ARM's requirement that the return type of an overriding function must be the same as the return type of the function it

PROVIDES ESSENTIAL HARD REAL-TIME RESPONSE

NEW
Version 2.0

General Software's

EMBEDDED

DOS™

REAL-TIME DOS FOR EMBEDDED SYSTEMS

SAVE UP TO 80% ON DOS ROYALTIES!

BUILT-IN REAL-TIME KERNEL

BOOTS FROM ROM OR DISK

SOURCE CODE AVAILABLE

USE BORLAND OR MICROSOFT TOOLS FOR DEVELOPMENT

SUGAR FREE

NO CHOLESTEROL

LOW SODIUM

NO ARTIFICIAL FLAVORS OR COLORS

FREE

CALL FOR A FREE BOOTABLE DEMO!

TEL (206) 391-4285
FAX (206) 557-0736
BBS (206) 557-4227

GENERAL SOFTWARE
Box 2571 • Redmond, WA 98073

Copyright (C) 1993-1994 General Software, Inc. General Software, the GS logo, and Embedded DOS are trademarks of General Software, Inc.

◆ Request 320 on Reader Service Card ◆

ADVANCED. SOLUTIONS FOR C & C++ PROGRAMMERS

The **C** Users Journal™

**If you're reading this journal,
you're among a select group
of advanced programmers
worldwide who program
professionally in C and C++.**

**Come meet
fellow programmers,
writers, and
C USERS JOURNAL
representatives at the**

**SOFTWARE
DEVELOPMENT
'94**

MARCH 15 - 17

**SAN JOSE
CALIFORNIA**

**See us at
Booth 1338.**

For more information call:
913-841-1631

We look forward to seeing you there!

overrides apparently doesn't pose any problems when dealing with pointers (or references) to objects at the root of the hierarchy, as in Listing 2. However, it often necessitates using casts when dealing with pointers to objects of other types in the hierarchy. For example, you cannot write

```
rectangle *r;
...
rectangle *cr = r->clone();
```

Listing 1 continued

```
private:
    double radius;
};

shape *circle::clone() const
{
    return new circle(*this);
}

...

//
// class 'rectangle' derived from 'shape'
//
class rectangle : public shape
{
public:
    rectangle(palette c, double h, double w);
    double area() const;
    shape *clone() const;
    const char *name() const;
    ostream &put(ostream &os) const;
private:
    double height, width;
};

shape *rectangle::clone() const
{
    return new rectangle(*this);
}

...

//
// class 'triangle' derived from 'shape'
//
class triangle : public shape
{
public:
    triangle(palette c, double s1, double s2, double a);
    double area() const;
    shape *clone() const;
    const char *name() const;
    ostream &put(ostream &os) const;
private:
    double side1, side2, angle;
};

shape *triangle::clone() const
{
    return new triangle(*this);
}

...

// End of File
```

because `r->clone()` returns a *shape* *. Even though a *rectangle* is a *shape*, a *shape* is not necessarily a *rectangle*. Therefore, you must add a cast, as in

```
rectangle *cr = (rectangle *)r->clone();
```

Casts are dangerous things. They tell the compiler to stop complaining and let you do what you want to do, or at least, what you think you want to do. But compilers are right more often than we

Listing 2 A crude application of the shape hierarchy in Listing 1

```
//
// 'clone_all' clones shape array 'sa1' with 'n'
// elements
//
shape **clone_all(shape *sa1[], size_t n)
{
    shape **sa2 = new shape *n;
    for (size_t i = 0; i < n; ++i)
        sa2[i] = sa1[i]->clone();
    return sa2;
}

//
// 'largest' returns the shape with the largest
// area from shape array 'sa' with 'n' elements
//
shape *largest(shape *sa[], size_t n)
{
    shape *s = 0;
    double m = 0;
    for (size_t i = 0; i < n; ++i)
        if (sa[i]->area() > m)
        {
            m = sa[i]->area();
            s = sa[i];
        }
    return s;
}

int main()
{
    const int N = 4;
    shape *s[N];
    shape *ls;
    s[0] = new circle(shape::RED, 2);
    s[1] = new triangle(shape::BLUE, 5, 6, asin(0.8));
    s[2] = new rectangle(shape::RED, 3, 4);
    s[3] = new circle(shape::GREEN, 3);
    cout << "The shapes are:\n";
    for (int i = 0; i < N; ++i)
        cout << i << "\t" << *s[i] << '\n';
    cout << '\n';
    shape **cs = clone_all(s, N);
    cout << "The cloned shapes are:\n";
    for (i = 0; i < N; ++i)
        cout << i << "\t" << *cs[i] << '\n';
    cout << '\n';
    ls = largest(cs, N);
    cout << "The shape with the largest area is a...\n";
    cout << *ls << ".\n";
    cout << "Its area is " << ls->area() << ".\n";
    return 0;
}

// End of File
```

care to admit. Casts indicate that you are doing something that is generally unsafe. Thus, you really should avoid casts in your C++ programs, probably even more so than in C programs. When casts are rare, the few casts you really do need will stand out and draw more scrutiny, which they deserve. (For more about avoiding casts, see Plum and Saks [1991].)

Of course, you don't really need a cast in the previous example, because you don't really need to call `clone` to replicate a *shape* that you know is a *rectangle*. Rather, you can simply `clone *r` with

```
rectangle *cr = new rectangle(*r);
```

But consider what happens if *rectangle* and *triangle* (and any other polygons) are derived from an abstract base class *polygon* derived from *shape*, rather than directly from *shape*, as outlined in Listing 3. (An abstract base class is a class with at least one *pure virtual* function. An abstract base class can be a derived class.)

To satisfy the ARM, all the clone functions in Listing 3 return a *shape* *. (The declaration of *polygon*'s clone function is inside a comment because you don't really need it. A function declared *pure virtual* in a base class remains *pure virtual* in the derived class unless overridden with an impure declaration.) When you clone a *polygon*, you get a pointer of type *shape* *, even though you know that pointer specifically addresses a *polygon*. Thus, you cannot clone a *polygon* and copy the result to a *polygon* * without a cast. That is, you cannot omit the cast in

RS232-Toolkit, SuperCom 2.1 for DOS, Windows and OS/2

for MS C/C++, Visual C++, Turbo/Borland C/C++, Turbo/Borland Pascal, Visual Basic

SuperCom is the development tool for exacting serial communication software. That means high data security and highest transmission speed. The SuperCom libraries are fast even in a multi-tasking operating system like Windows or OS/2.

The same programming interface is used among different languages and operating systems.

- Interrupt driven: transmission, reception, linestatus, modem status. Up to 115,200 bps.
- UARTS: 8250, 16450, 16550 FIFO, any port address.
- Simultaneous COM_1..COM_32 (and unlimited).
- Direct register programming, Interrupt-Sharing.
- Flow control: RTS/CTS, DTR/DSR, XON/XOFF and user defined.
- Protocols: ASCII, XMODEM, XMODEM/CRC.
- Timer, Ctrl-Break and Exception handling.
- Multitasking support (Windows, OS/2)
- Protected Mode Interface, 386-Technology.
- Language independent DLL (Windows, OS/2) which can be used for simultaneous transfers by applications.
- Multiserial board support (AST, ARNET, DigiBoard PC/X, STARGATE) Reduces loading of CPU through support of intelligent DigiBoard PC/Xe, PC/Xi boards.
- Modem support.
- No resident drivers. Just link the LIB to your Program.
- No Royalties.
- FREE technical support.
- Full source code (C or Pascal and optimized ASM).
- DLL interface for VisualBasic (Windows) included.
- SuperCom++ (C++ or Pascal OOP) included.
- Protected Mode Interface (Windows) included.
- C package (MS C/C++, Borland C/C++ VisualBasic).
- Pascal package (Turbo Pascal, VisualBasic).

C/C++ or Pascal package for DOS	\$320
C/C++ or Pascal package for Windows	\$430
C/C++ or Pascal package for DOS + Windows	\$570
C/C++ package for OS/2	\$430
C/C++ package for DOS+Windows+OS/2	\$880

VISA, MC, COD, Check accepted.

Support for Windows NT coming soon

ADONIS Micro-Software
Hoelderlinstr. 32
D-75433 Maulbronn, Germany
Phone: 49-7043-40449
FAX: 49-7043-40440

Fine Line International
7000 Malone Rd, Forestville
CA 95436, USA
Phone: 1-707-887-3400
FAX: 1-707-887-1015



All of the casts in the previous examples cast pointers to base class objects into pointers to derived class objects. These casts are commonly called "downcasts" because most people draw class hierarchies with the base classes above their derived classes. Remember, when a class *D* is publicly derived from *B*, a *D* is a *B*, so you can safely convert a *D* * to a *B* * without a cast. But a *B* * is not necessarily a *D* *, so you can't convert a *B* * downward to a *D* * without a cast. Thus, like all other casts, downcasts are generally unsafe unless you're absolutely, positively sure that your *B* * actually points to a *D*.

But the downcast in

```
polygon *cp = (polygon *)p->clone();
```

is actually quite safe because we do know that *p->clone()* returns a *polygon* *. In fact, there's a whole family of similarly safe downcasts that occur commonly in object-oriented systems. The problem is that, on the surface, the safe casts look just like the unsafe ones. The cure is to augment the rules for virtual overriding so that you can write the safe conversions without casts.

For example, you should be able to declare each virtual clone function in a hierarchy so that it returns a pointer whose static type is the same as its dynamic type. That is, *circle::clone*

a cast. For instance, given

Listing 3 The shape hierarchy with rectangle and triangle derived from abstract base class polygon

```
class shape
{
public:
    ...
    virtual shape *clone() const = 0;
    ...
};

...

class circle : public shape
{
public:
    shape *clone() const;
    ...
};

shape *circle::clone() const
{
    return new circle(*this);
}

...

class polygon : public shape
{
public:
    // shape *clone() const = 0; // still pure
    ...
};

...

class rectangle : public polygon
{
public:
    shape *clone() const;
    ...
};

shape *rectangle::clone() const
{
    return new rectangle(*this);
}

...

class triangle : public polygon
{
public:
    shape *clone() const;
    ...
};

shape *triangle::clone() const
{
    return new triangle(*this);
}

...

// End of File
```

CAD-UL

Improve the productivity
of your cross software
development with the

Organon

Cross Compiler

for the Intel 386/486

Choose this complete toolset:
compiler, assembler, and linker
all from the same company.
Specially designed for
embedded development and
the Intel 386/486.

Organon Compiler cc386:
MS-DOS, MS-Windows \$ 1950
Workstation \$ 3950

Visa and MasterCard
accepted

Computer Aided
Design Ulm GmbH
P.O. BOX: 1280
D-89002 Ulm, Germany
Tel. +49 731 937600

CompuServe: Peter Horn,
ID100273,305

Fax +49 731 9376027
for free information!

- Compatible with Intel ic386 (e.g. memory models, programs, object format).
- For use with C++ and ANSI C.
- Optional: Our powerful high-level language debugger Organon XDB 386 for ROM monitor and RT kernels.
- Available for workstations, MS-DOS, and MS-Windows.

All trademarks owned by their
respective companies.

◆ Request 209 on Reader Service Card ◆


```
shape *s;
circle *c;
```

you can write

```
shape *cs = s->clone();
```

to clone a *shape*, and

```
circle *cc = c->clone();
```

to clone a *circle*.

In a sense, declaring *circle::clone* to return a *circle ** doesn't introduce any new conversions. It merely shifts the exact point where the conversions occur, or eliminates the conversions altogether. For example, when you write *circle::clone* as

```
shape *circle::clone() const
{
    return new circle(*this);
}
```

a conversion from *circle ** to *shape ** occurs as part of the return statement. Then there's no conversion at all in a calling expression like

```
shape *s;
...
shape *cs = s->clone();
```

In contrast, when you write *circle::clone* as

```
circle *circle::clone() const
{
    return new circle(*this);
}
```

no conversion occurs inside the function, but an implicit conversion from *circle ** (or *rectangle ** or *triangle **) to *shape ** occurs in the calling expression. The net effect is the same.

The New, Relaxed Rules

The C++ standards committee agreed that the ARM's requirement on the return type of virtual functions is a bit too restrictive. Thus, the current draft of the Working Paper (the standard-to-be) relaxes the original rule. The new rule as it appears in the Working Paper is jargon-rich and seems to change with each new draft, so I'll spare you the exact words. Here's more-or-less what it says:

For all classes *B* and *D* defined as

```
class B
{
    ...
    virtual BT f();
    ...
};
```

```
class D : public B
{
    ...
}
```



PC-lint for C/C++
presents Bug # 1733

```
#include <stdio.h>

class X
{
public:
    int *px;
    X( int init )
    { px = new int; *px = init; }
    ~X() { delete px; }

void print( X x )
{ printf( "%d\n", *x.px ); }

int main() {
    X x(15); print( x );
    X y(16); print( x ); print( y );
    return 0;
}
```

The output the programmer expected to see was 15, 15, and 16. Instead, he got 15, 16, and 16. What went wrong? Call if you need a hint. Refer to Bug #1733.

PC-lint for C/C++ will catch this and many other bugs. It will analyze a mixed suite of C and C++ modules to uncover bugs, glitches, quirks and inconsistencies.

Numerous C++ Warnings and Messages: Are your inherited destructors virtual? Are your constructor *new*'s matched by your destructor *delete*'s? Are your initializers in order? Are names inadvertently hiding other names? Are your C++ modules consistent with your C modules? Much, much, more.

Plus Our Traditional C Warnings:

Uninitialized variables, unaccessed variables, possibly uninitialized variables, strong type mismatches, indentation irregularities, loss of precision, strange uses of Booleans, signed/unsigned mismatches, suspicious expressions, unused macros, etc. etc.

Full C++ Support - PC-lint for C/C++ is based on the ARM and is tracking the latest ANSI/ISO draft including exceptions and templates. It supports both Borland and Microsoft C/C++.

Options Galore: A plethora of options for message suppression, message format, compiler dependencies, etc. All messages can be individually suppressed or enabled, both locally and globally. Numerous compilers/libraries supported. Runs on MS-DOS (Optional built-in 386 DOS extender) and OS/2.

PC-lint for C \$139
PC-lint for C/C++ \$239

PC-lint users: call for update pricing
Unix and Mainframe programmers: call for pricing for FlexLint.

Gimpel Software

3207 Hogarth Lane, Collegeville, PA 19426
CALL TODAY (215) 584-4261 Or FAX (215) 584-4266
30 Day Money-back Guarantee.

PA add 6% sales tax.

PC-lint and FlexLint are trademarks of Gimpel Software

```
DT f();
...
};
```

types *BT* and *DT* must be identical, or they must satisfy either of the following conditions:

1. *BT* is *BB ** and *DT* is *DD ** where *DD* is derived from *BB*.
 2. *BT* is *BB &* and *DT* is *DD &* where *DD* is derived from *BB*.
- In either case (1) or (2),
3. class *D* must have the access rights to convert a *BB ** (or *BB &*) to a *DD ** (or *DD &*, respectively).

In most common applications, *BB* is a public base class of *DD*, so *D* can perform the conversions. But, for example, if *BB* is a private base class of *DD* then the conversions are not valid, and *BT* and *DT* will not satisfy condition (3).

The above rules apply even if *D* is derived indirectly from *B*. Or, *BB* might be *B* and *DD* might be *D*. The latter, in fact, is the case with clone functions.

Listing 4 shows the shape hierarchy of Listing 1 rewritten using the new relaxed rules for the return type of virtual functions. For completeness, I've included all the member function bodies so you can use them to build and execute the test code in Listing 2.

Although the committee adopted these relaxed rules in March, 1992, I believe most vendors have yet to release a C++ compiler that supports them. As of early 1994, only two of the six PC-based compilers I own (Borland 4.0 and Watcom 9.5) can compile Listing 4 without error.

cv-qualifiers in Return Types

According to the current (September 1993) Working Paper, the cv-qualifiers (*const* and *volatile*) in the return types of the overriding and overridden functions need not be identical. My understanding is that the overriding function's return type cannot have

Listing 4 The shape class hierarchy with virtual cloning functions employing the relaxed return type rules

```
#include <iostream.h>
#include <math.h>
#include <stddef.h>

//
// base class 'shape'
//
class shape
{
public:
    enum palette { BLUE, GREEN, RED };
    shape(palette c);
    virtual double area() const = 0;
    virtual shape *clone() const = 0;
    palette color() const;
    virtual const char *name() const = 0;
    virtual ostream &put(ostream &os) const;
private:
    palette _color;
    static const char *color_image[RED - BLUE + 1];
};

const char *shape::color_image[shape::RED - shape::BLUE + 1] =
{ "blue", "green", "red" };

shape::shape(palette c) : _color(c) { }

ostream &shape::put(ostream &os) const
{
    return os << color_image[_color] << ' ' << name();
}

shape::palette shape::color() const
{
    return _color;
}

//
// class 'circle' derived from 'shape'
//
const double pi = 3.1415926;


class circle : public shape
{
public:
    circle(palette c, double r);
    double area() const;
    circle *clone() const;
    const char *name() const;
    ostream &put(ostream &os) const;
private:
    double radius;
};

circle::circle(palette c, double r) : shape(c), radius(r) { }

double circle::area() const
{
    return pi * radius * radius;
}
```

Windows Programming Training

- ◆ Learn SDK
- ◆ Intensive 4 or 5 day classes
- ◆ Fundamentals & Advanced
- ◆ Using C, C++, or Visual Basic
- ◆ C, C++ and Visual Basic Training Available
- ◆ Borland or Microsoft Visual C++ Compilers
- ◆ Staffed by Professional Programmer/Trainers
- ◆ Public Classes in Center City, Philadelphia
- ◆ On-Site Training Available at Your Location
- ◆ Programmer Training Since 1985
- ◆ Call for Brochure



COMPUTER LANGUAGE ARTS
P.O. Box 3733 • Cherry Hill • NJ • 08034
800-377-6700

◆ Request 440 on Reader Service Card ◆

Listing 4 *continued*

```

const char *circle::name() const
{
    return "circle";
}

ostream &circle::put(ostream &os) const
{
    return shape::put(os) << " with radius = " << radius;
}

circle *circle::clone() const
{
    return new circle(*this);
}

//
// class 'rectangle' derived from 'shape'
//
class rectangle : public shape
{
public:
    rectangle(palette c, double h, double w);
    double area() const;
    rectangle *clone() const;
    const char *name() const;
    ostream &put(ostream &os) const;
private:
    double height, width;
};

rectangle::rectangle(palette c, double h, double w)
    : shape(c), height(h), width(w) { }

double rectangle::area() const
{
    return height * width;
}

const char *rectangle::name() const
{
    return "rectangle";
}

ostream &rectangle::put(ostream &os) const
{
    return shape::put(os) << " with height = " << height
        << " and width = " << width;
}

rectangle *rectangle::clone() const
{
    return new rectangle(*this);
}

//
// class 'triangle' derived from 'shape'
//
class triangle : public shape
{
public:
    triangle(palette c, double s1, double s2, double a);
    double area() const;
    triangle *clone() const;
    const char *name() const;
    ostream &put(ostream &os) const;
private:
    double side1, side2, angle;
};

triangle::triangle(palette c, double s1, double s2, double a)
    : shape(c), side1(s1), side2(s2), angle(a) { }

double triangle::area() const
{
    return side1 * sin(angle) * side2 / 2;
}

const char *triangle::name() const
{
    return "triangle";
}

ostream &triangle::put(ostream &os) const
{
    return shape::put(os) << " with one side = " << side1
        << ", another side = " << side2
        << " and angle = " << angle;
}

triangle *triangle::clone() const
{
    return new triangle(*this);
}

ostream &operator<<(ostream &os, const shape &s)
{
    return s.put(os);
}

// End of File

```

SQL DATABASE ENGINE FOR C PROFESSIONALS

*Give your programs a powerful database at the lowest cost
with the Just Logic database engine.*

Just Logic is meant for professionals who write simple to very complex C/C++ systems and who need a true database without compromising performance.

The Just Logic database engine delivers:

- A clear and easy to read manual
- Comprehensive examples
- ANSI SQL
- Compatibility with popular compilers
- C and C++ interfaces
- RUNTIME LICENCE INCLUDED
- 30-day risk free money back guarantee

Limited time offer

DOS and Windows version **\$99**

OS/2 version **\$99**

DOS, Windows and OS/2 version **\$149**

New products

SCO Unix **\$595**

BSD Unix **\$295**

Coherent **\$129**

Call now: (800) 267-6887 (toll-free USA and Canada)
(800) 234-0141 (InstantInfo Doc #1185)
(514) 761-6887 (International)
(514) 642-6480 (Fax)

JUST LOGIC
TECHNOLOGIES

P.O. Box 63050,
40 Commerce St.,
Nun's Island, Que, Canada,
H3E 1V6

◆ Request 295 on Reader Service Card ◆

any cv-qualifiers that are not also in the overridden function's return type. Listing 5 shows some examples.

Class *B* in Listing 5 declares virtual function *f* with a return type *const BB **, but class *D* overrides it with a function that returns *DD ** (where *DD* is publicly derived from *BB*). Hence, if *bp* is a *B ** that actually points to a *D*, then

```
const BB *bbp = bp->f();
```

invokes *D::f* applied to **b*. *D::f* returns a pointer to a non-const *DD* object, which the expression *bp->f()* quietly converts to *const BB **.

Pointer conversions that add cv-qualifiers are always safe, but conversions that strip off cv-qualifiers are not. Thus, given

```
char *cp;
const char *ccp;
```

then

```
ccp = cp;
```

is safe, but

```
cp = ccp;
```

is not. Similarly, as you convert derived types to their base types, adding cv-qualifiers to pointer types should not make the conversions any less safe.

Listing 5 also shows that derived class *D* has a virtual function *g* returning a *const DD &* that overrides a function returning a *const volatile BB &*. This is also valid. However, if you omit *volatile* from the return type of *B::g*, then *D::g* is erroneous.

None of the compilers I own support this feature yet.

More to Come

Over the past two years, the standards committee has added several other new features to C++:

- run-time type identification
- mutable members for *const* objects
- namespaces
- a predefined boolean type
- new syntax for casts

I will explain them all in upcoming columns.

Meeting Dates, Etc.

WG21+X3J16 will meet three times in 1994:

- March 6-11 in San Diego, CA USA
- July 10-15 in Waterloo, Ontario Canada
- November 6-11 in Valley Forge, PA USA

If all goes as scheduled, the draft standard should be available for public review and comment shortly after the July meeting.

If you would like to participate in the standards process as a member of X3J16, contact the vice-chair:

Josée Lajoie
IBM Canada Laboratory
844 Don Mills Rd.
North York, Ontario M3C 1V7 Canada
(416)448-2734
josee@vnet.ibm.com

References

Ellis and Stroustrup [1990]. Margaret A. Ellis and Bjarne Stroustrup. *The Annotated C++ Reference Manual*. Addison-Wesley.

Plum and Saks [1991]. Thomas Plum and Dan Saks. *C++ Programming Guidelines*. Plum Hall.

debug RS-232

Record and analyze serial communications with your PC!

MicroTAP™ transforms your PC into a serial line monitor. Save time, eliminate frustration, and put your PC investment to work! End serial communications guesswork—now you can prove and document who sent what and when with *microsecond accuracy*.

Featuring: Up to four user-alterable, multitasking display windows; Microsecond data and signal event resolution; Editable EGA/VGA fonts; 32K-64M log files; Macro keystroke and display recording; PostScript and ASCII export; Laptop support; All hardware baud rates (2-115,200); and Context-sensitive Hypertext help. Ultra-fast displays. Cable set included. Satisfaction guaranteed!



v3.0 only
\$349!

"It's a very complete package that answers all my needs."

— Roger Boots
Eastman Kodak

Pays for itself immediately! Order line: 1-800-397-1368

PALADIN SOFTWARE, INC. 3945 KENOSHA AVE. • SAN DIEGO, CA 92117
"In real-time since 1982" TEL: (619) 490-0368 • FAX: (619) 490-0177

◆ Request 227 on Reader Service Card ◆

Listing 5 A derived class whose virtual functions have fewer cv-qualifiers than the functions they override

```
class B
{
...
virtual const BB *f();
virtual const volatile BB &g();
...
};

class D : public B
{
...
DD *f();
const DD &g();
...
};

// End of File
```

The Preprocessor

To use C effectively you really have to master two languages: the C language proper and the preprocessor. Before a compiler begins the usual chores of syntax checking and instruction translation, it submits your program to a preliminary phase called preprocessing, which alters the very *text* of the program according to your instructions. The altered text that the compiler sees is called a *translation unit*. In particular, the preprocessor performs the following three functions for you:

- 1) header/source file inclusion
- 2) macro expansion
- 3) conditional compilation

In this article I will illustrate these features of the preprocessor.

The Include Directive

One of the first source lines any C programmer sees or composes is this:

```
#include <stdio.h>
```

Take a moment right now and jot down everything you know about this statement.

Let's see how you did. *stdio.h* is of course a standard library header, so called because such include directives usually appear near the beginning of a source file so that their definitions will be in force throughout the rest of the compilation. We commonly think of it as a header *file*, but there is no requirement that the definitions and declarations pertaining to standard input and output reside in a file. The C Standard only requires that these definitions and declarations replace the include directive in the text of the program before translation. They could reside in tables internal to the preprocessor. Most compiler implementations do supply header files for the standard library, however. MS-DOS compilers install header files in a suitable subdirectory. Here is a sampling:

```
\BC4\INCLUDE      /* Borland C++ 4.0 */
\MSVC\INCLUDE     /* Microsoft Visual C++ */
\WATCOM\H         /* Watcom C/C++ */
```

On UNIX systems you will find header files in */usr/include*.

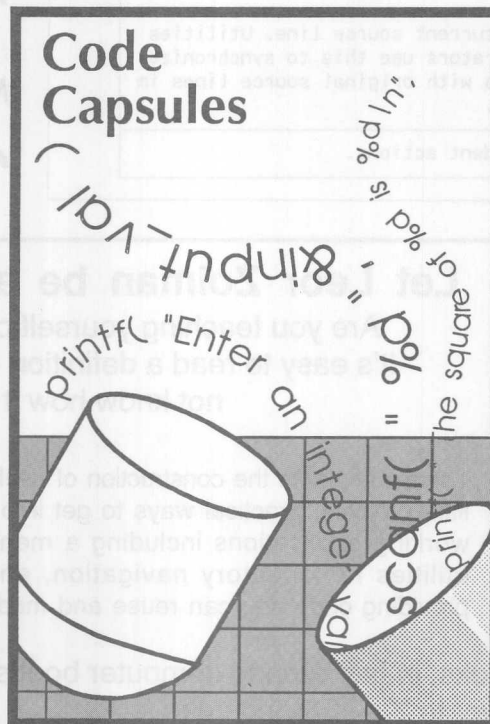
Since an implementation is not even obliged to supply headers in the form of physical files, it's no surprise that those implementations providing files don't always give them the same name as the include directive. After all, how could a compiler supply a file named *stdio.h* on a platform whose file system didn't allow periods in a file name? On MS-DOS systems there can be no file that exactly matches the C++ header *<strstream.h>*, because the file system only allows up to eight characters before the period.

Most MS-DOS implementations map header names into file names by truncating the base part (the portion before the period) to eight characters, and the extension to three (so the definitions for *<strstream.h>* reside in the file *STRSTREA.H*). A standard-conforming implementation must supply a mapping to the local file system for user-defined header names having at least six characters before the period and one character after.

Conforming compilers also support include directives with string-like arguments, as in:

```
#include "mydefs.h"
```

The string must represent a name recognized by the local file system. The file must be a valid C/C++ source file and, like the standard headers, usually contains function prototypes, macro definitions, and other declarations. An implementation must specify the mechanism it uses to locate the requested source file. On platforms with hierarchical file systems, the compiler usually searches the current directory first. If that fails, it then searches the subdirectory



Chuck Allison is a regular columnist with CUJ and a software architect for the Family History Department of the Church of Jesus Christ of Latter Day Saints Church Headquarters in Salt Lake City. He has a B.S. and M.S. in mathematics, has been programming since 1975, and has been teaching and developing in C since 1984. His current interest is object-oriented technology and education. He is a member of X3J16, the ANSI C++ Standards Committee. Chuck can be reached on the Internet at allison@decus.org, or at (801)240-4510.

reserved for the standard headers. Because standard header names are special preprocessing tokens and not strings, any backslashes in a header name are not interpreted as escape characters. In the following directive, a double backslash is not needed.

```
#include <sys\stat.h>          /* \, not \\ */
#include "\\project\\include\\mydefs.h"
```

Included files may themselves contain other include directives, nested up to eight levels deep. Since some definitions (like typedefs) must only appear once during a compilation, you must guard against the possibility of a file being included more than once. The customary technique defines a symbol associated with the file. Exclude the text of the file from the compilation if the symbol has already been seen by the compiler, as in the following:

```
/* mydefs.h */
#ifndef MYDEFS_H
#define MYDEFS_H

<declarations/definitions go here>

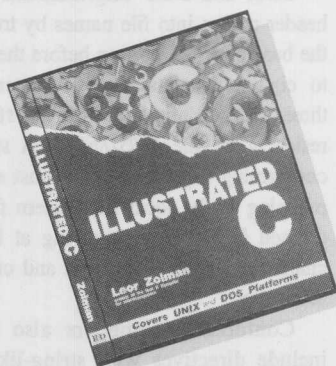
#endif
```

Macros

As you can see, there's more to the `#include` directive than meets the eye. C

Table 1 *Preprocessor directives*

<code>#include</code>	Includes text of header or source file.
<code>#define</code>	Enters a symbol into the symbol table for the current compilation unit, with an optional value.
<code>#undef</code>	Removes a symbol from the symbol table.
<code>#if</code> <code>#elif</code> <code>#else</code> <code>#endif</code>	Control flow directives for conditional compilation.
<code>#ifdef</code> <code>#ifndef</code>	Symbol table query directives. (Also used for conditional compilation).
<code>#line</code>	Renumbers the current source line. Utilities like code generators use this to synchronize generated lines with original source lines in error messages.
<code>#pragma</code>	Compiler-dependent actions.



Let Leor Zolman be a personal mentor.

Are you teaching yourself or someone else C?
It's easy to read a definition of a function but still
not know how it is used!

Zolman explores the construction of useful applications from start to finish; little projects, practical ways to get into self-taught C. He explains eight working applications including a menuing system, a mini-database, utilities for directory navigation, and a portable journal manager; providing code you can reuse and modify for your own projects.

"This is one of the better C books, in fact among computer books
it has been my favorite." — Fredric Luke

"Excellent programming book, far better than most. I have used
several of the routines as the basis for my own developments.
I was very impressed with the down to earth explanations."
— John Moran

Only
\$29.95
(\$39.95 with disk)

Identify Source Code **ZOLC1** to Order **W34** for the book or **W36** for book with disk.



1601 W. 23rd., Suite 200, Lawrence, KS 66046
To Order Call 913-841-1631 FAX 913-841-2624



provides eleven other preprocessor directives you can use to alter your source text in meaningful ways (see Table 1). (All begin with the '#' character, which must be the first non-space character on its source line.) In this section I elaborate on one of the other directives, the `#define` directive, to introduce a very useful construct called a *macro*.

The `#define` directive creates macro definitions. A macro is a name for a sequence of zero or more preprocessing tokens. (Valid preprocessing tokens include valid C language tokens such as identifiers, strings, numbers and operators; and any single character). For example, the line

```
#define MAXLINES 500
```

associates the text `500` with the symbol `MAXLINES`. The preprocessor keeps a table of all symbols created by the `#define` directive, along with the corresponding replacement text. Whenever the preprocessor encounters the token `MAXLINES` outside of a quoted string or comment, it replaces `MAXLINES` with the token `500`. In later phases of compilation it appears as if you actually typed `500` instead of `MAXLINES`. It is important to remember that this operation consists of mere *text replacement*. No semantic analysis occurs during preprocessing.

A macro without parameters, such as `MAXLINES`, is sometimes called an *object-like* macro because it defines a program constant that looks like an object. Because object-like macros are often constants, it is customary to type them in upper case as a hint to the reader. You can also define *function-like* macros with zero or more parameters, as in the following code fragment:

```
#define beep()    putc('\a',stderr)
#define abs(x)    ((x) >= 0 ? (x) : -(x))
#define max(x,y)  ((x) > (y)) ? (x) : (y)
```

There must be no whitespace between the macro name and the first left parenthesis. The expression

```
abs(-4)
```

expands to

```
((-4) >= 0 ? (4) : (-(-4)))
```

You should always parenthesize macro parameters (such as `x`) in the replacement text. This practice prevents surprises from unexpected precedence in complex expressions. For example, if you had used the following naive mathematical definition for absolute value:

```
x >= 0 ? x : -x
```

then the expression `abs(a - 1)` would expand to

```
a - 1 >= 0 ? a - 1 : -a - 1
```

which is incorrect when `a - 1 < 0` (it should be `-(a - 1)`).

Even if you put parentheses around all arguments, you should usually parenthesize the entire replacement expression as well to

avoid surprises with respect to the surrounding text. To see this, define `abs()` without enclosing parens, as in:

```
#define abs(x) (x) >= 0 ? (x) : (-x)
```

Then `abs(a) - 1` expands to

```
(a) >= 0 ? (a) : -(a) - 1
```

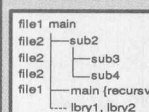
Table 2 Pre-defined macros

Macro	Value
<code>__LINE__</code>	The number of the current source line (equal to one more than the number of newline characters read so far).
<code>__FILE__</code>	The name of the source file.
<code>__DATE__</code>	The date of translation, in the form "Mmm dd yyyy."
<code>__TIME__</code>	The time of translation, in the form "hh:mm:ss".
<code>__STDC__</code>	1, if the compilation is in "standard" mode.

C AND C++ DOCUMENTATION

C-METRIC™ (\$59) COMPLEXITY / QUALITY

- Calculates "cyclomatic" path complexity for functions and system
- Counts lines with comments, code, and 'C' statements

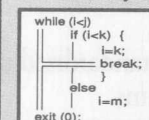


C-CALL™ (\$69) FUNCTION HIERARCHY

- Tree-Diagram showing function hierarchy
- Table-Of-Contents of functions versus files
- Summary and detailed cross-reference of functions

C-CMT™ (\$69) FUNCTION COMMENT

- Generates and inserts function comment blocks
- Can be re-run to update the comment blocks
- Retains any user-generated comments



C-LIST™ (\$69) LISTS OR REFORMATS

- Action-Diagrams show logic/control flow
- Optional line numbers, page numbers, and titles
- Reformats source to various standardized formats

C-REF™ (\$59) CROSS-REFERENCES IDENTIFIERS

- Local/parameter/global/define summary or cross-reference
- Produces class-hierarchy tree-diagram for C++ classes

SPECIAL: (\$199) C-DOC™ DOS Package (\$325 Value)

- All 5 programs fully integrated as 1 overall C-DOC program
- Processes multiple directories/files up to 15,000 total lines
- Unconditional 30-day money-back guarantee of satisfaction

NEW: C-DOC™ Professional (\$299): DOS, OS/2, Windows

- Processes 150,000 lines, 3-ring binder/case, 2-pass deferred reports

!! NEW 5.0 !! Point-and-click G.U.I. operation !!

SOFTWARE BLACKSMITHS INC

6064 St Ives Way, Mississauga
ONT Canada L5N-4M1

VISA MasterCard
Voice/Fax: (416)-858-4466
Demo/BBS: (416)-858-1916

◆ Request 147 on Reader Service Card ◆

It is also dangerous to use expressions with side effects as macro arguments. For example, the macro call `abs(++i)` expands to

```
((-i++) >= 0 ? (i++) : (-i++))
```

Listing 1 Prints the pre-defined macros

```
/* sysmac.c: Print system macros */
#include <stdio.h>

main()
{
    printf(" _DATE_ == %s\n", _DATE_);
    printf(" _FILE_ == %s\n", _FILE_);
    printf(" _LINE_ == %d\n", _LINE_);
    printf(" _TIME_ == %s\n", _TIME_);
    printf(" _STDC_ == %d\n", _STDC_);
    return 0;
}

/* Output:
_DATE_ == Dec 18 1993
_FILE_ == sysmac.c
_LINE_ == 9
_TIME_ == 19:05:06
_STDC_ == 1
*/
/* End of File */
```

probably isn't what you had in mind.

Pre-defined Macros

Conforming implementations supply the five built-in object-like macros shown in Table 2. The last three macros remain constant during the compilation of a

Listing 2 Illustrates an assertion failure

```
/* fail.c */
#include <stdio.h>
#include <assert.h>

main()
{
    int i = 0;

    assert(i > 0);
    return 0;
}

/* Sample Execution:
C:>assert
Assertion failed: i > 0,
    file assert.c, line 9
Abnormal program termination
*/
/* End of File */
```

leading underscore followed by either an uppercase letter or another underscore.

You may not redefine any of these five macros with the `#define` directive, nor remove them with the `#undef` directive. Most compilers support multiple modes, some of which are not standard-conforming. (To guarantee that the sample program in Listing 1 will run correctly under Borland C, for example, you need to run in "ANSI mode" via the "-A" command-line option.)

Conforming compilers also provide a function-like macro, `assert`, which you can use to put diagnostics in programs. If its argument evaluates to zero, `assert` prints the argument along with source file name and line number (using `_FILE_` and `_LINE_`) to the standard error device and aborts the program (see Listing 2). For more information on using the `assert` macro, see the Code Capsule "File Processing, Part 2" in the June 1993 issue of *CUJ*.

A compiler is allowed to provide macro versions for any functions in the standard library (`getc` and `putc` usually come as macros for efficiency). With the exception of a handful of required function-like macros (`assert`, `setjmp`, `va_arg`, `va_end`, and `va_start`), an implementation must also supply true function versions for all functions in the standard library. A macro version of a library function in effect hides its prototype from the compiler, so its arguments are not type-checked during translation. To force the true function to be called, remove the macro definition with the `#undef` directive, as in

```
#undef getc
```

Alternatively, you can surround the function name in parentheses when you call it, as in:

```
c = (getc)(stdin);
```

There's no danger of this expression matching the macro definition since a left parenthesis does not immediately follow the function name.

Client/Server is the Future

Start today. Send for information on Emerald Bay DBMS Toolkits for Xbase, C and Pascal programmers.

See why companies like Quarterdeck have standardized on Emerald Bay for their own in-house enterprise-wide systems.

Emerald Bay solutions are affordable—thousands less than equivalent SQL systems from mega publishers. Vulcan (our version of Xbase) interpreter and compiler is only \$395. The Pascal version is \$195 and the C version in \$395. Each Toolkit includes a 2-user Database Server. 4-user servers are \$195, 8 users \$395 and 100 users \$595.

And Emerald Bay solutions are easy to implement, since they build on your knowledge of Clipper, dBASE, DBXL, Foxbase and Quicksilver. Call or write for information today. The future won't wait.

Emerald Bay Literature & Orders (800)354-3222 Ext. E-01
(818)248-0877 Fax: (818)248-2605
2155 Verdugo Boulevard, Suite 20, Montrose, California 91020

◆ Request 296 on Reader Service Card ◆

Conditional Compilation

You can selectively include or exclude segments of code with conditional directives. For example, you can embed the following excerpt in your code to accommodate different syntaxes of the *delete* operator in earlier versions of C++:

```
#if VERSION < 3
    delete [strlen(p) + 1] p;
#else
    delete [] p;
#endif
```

Your compiler probably supplies a macro similar to *VERSION* (Borland C++ defines *__BCPLUSPLUS__*, Microsoft *_MSCVER*). The argument of an *#if* directive must evaluate to an integer constant, and obeys the usual C rule of non-zero means true, zero false. You cannot use casts or the *sizeof* operator in such expressions.

C++ implementations also pre-define the macro *__cplusplus*, which you can use to customize your code for mixed C/C++ environments. For example, if you want to link with existing C code in a C++ environment, you need to use the extern "C" linkage specification (which of course is not valid in a C environment). The following excerpt will do the right thing in either environment:

```
#ifdef __cplusplus
extern "C"
{
#endif

<put C declarations here>

#ifdef __cplusplus
#endif
```

The *#if* directive is handy when you want to comment out long passages of code. You can't just wrap such sections in a single, enclosing comment because there are likely to be comments in the code itself (right?), causing the outer comment to end prematurely. It is better to enclose the code in question in the body of an *#if* directive that always evaluates to zero:

```
#if 0
<put code to be ignored here>
#endif
```

Preprocessor Operators

Sometimes you just want to know if a macro is defined, without using its value. For example, if you only support two compilers, you might have something like the following in your code:

```
#if defined _MSCVER
<put Microsoft-specific statements here>
#elif defined __BCPLUSPLUS__
<put Borland-specific statements here>
#else
```

```
#error Compiler not supported.
#endif
```

defined is one of three preprocessor operators (see Table 3). The *defined* operator evaluates to 1 if its argument is present in the symbol table, meaning that the macro was either defined by a previous *#define* directive or the compiler provided it as a built-in macro. The *#error* directive prints its argument as a diagnostic and halts the translator.

Turn your C/C++ compiler into a powerful scientific programming tool...

C/Math Toolchest



Now you can develop sophisticated scientific and engineering applications using your favorite C or C++ compiler. With a broad range of mathematical functions, the C/Math Toolchest™ gives C as much number crunching power as FORTRAN. More than 135 functions provide support for:

- Numerical Analysis and Digital Signal Processing:** including integration, differentiation, interpolation, root finding, convolution, FFTs, power spectrum analysis, and filter design.
- Probability and Statistics:** including uniform, normal, binomial, Poisson, and hypergeometric distributions, and least-squares regression.
- Complex Arithmetic and Linear Algebra:** including a comprehensive set of vector and matrix operations, and solutions to simultaneous linear equations.

To graphically view the results of your number crunching, we've also included GRAFIX™, a graphical data analysis program. The GRAFIX™ program makes it easy to edit, plot, interpolate, or perform regression analysis of your data.

You may also purchase the C source code for the library and GRAFIX™ program. The C/Math Library Source works with any ANSI standard C or C++ compiler. The C/Math GRAFIX Source works with the C and C++ compilers from Mix®, Borland®, and Microsoft®. Prebuilt libraries for the DOS versions of these compilers are included with the C/Math Toolchest.

Now Only \$29.95!
Includes 430 page manual

Order now by calling our toll free number or mail the coupon to:

Mix Software
1132 Commerce Drive
Richardson, TX 75081

1-800-333-0330
60 Day Money Back Guarantee

Not Copy Protected ■ No Royalties
For technical support, please call:
1-214-783-6001
FAX: 1-214-783-1404

Order Coupon C

Name _____

Street _____

City _____

State _____ Zip _____

Telephone _____

☐ Send me free brochures for all Mix products

Paying By: ☐ Money Order ☐ Check

☐ Visa ☐ MC ☐ AmX ☐ Disc.

Card# _____

Exp. Date _____

Disk Size ☐ 5 1/4" ☐ 3 1/2"

Qty.	Product	Price	Subtotal
_____	C/Math Toolchest	\$29.95	_____
_____	C/Math Library Source	\$10.00	_____
_____	C/Math GRAFIX Source	\$10.00	_____

Add Shipping
(\$5 USA, \$10 Canada, \$20 Foreign) _____

Texas Residents Add 8.25% Sales Tax _____

Total Amount of Your Order _____

C/Math Toolchest and GRAFIX are trademarks of Mix Software, Inc. Mix, Borland, and Microsoft are trademarks of the respective companies.

◆ Request 236 on Reader Service Card ◆

It isn't necessary to assign a value to a macro. For example, to insert debug trace code into your program, you can do the following:

```
#if defined DEBUG
fprintf(stderr, "x = %d\n", x);
#endif
```

To define the `DEBUG` macro, just insert the following statement before the first use of the macro:

```
#define DEBUG
```

The following equivalences are recognized by the preprocessor:

```
#if defined X      <==>      #ifdef X
#if !defined X     <==>      #ifndef X
```

Using the `defined` operator is more flexible than the equivalent directives on the right because you can combine multiple tests as a single expression, as in:

```
#if defined __cplusplus && !defined DEBUG
```

The operator `#`, the "stringizing" operator, effectively encloses a macro argument in quotes. As the program in Listing 3 illustrates, stringizing can be useful for debugging. The `trace()` macro encloses its arguments in quotes so they become part of a `printf` format statement. For example, the expression `trace(i,d)` becomes

```
printf("i " = %" "d" "\n", i);
```

and, after the compiler concatenates adjacent string literals it sees this:

Listing 3 Illustrates the stringizing operator

```
/* trace.c: Illustrate a trace *
 * macro for debugging */

#include <stdio.h>

#define trace(x,format) \
    printf("#x " = %" #format "\n",x)

main()
{
    int i = 1;
    float x = 2.0;
    char *s = "three";

    trace(i,d);
    trace(x,f);
    trace(s,s);
    return 0;
}

/* Output:
i = 1
x = 2.000000
s = three
*/
/* End of File */
```

```
printf("i = %d\n", i);
```

There is no way to build quoted strings like this without the stringizing operator because the preprocessor ignores macros inside quoted strings.

The token-pasting operator, `##`, concatenates two tokens together to form a single token. The call `trace2(1)` in Listing 4 is translated into

```
trace(x1,d)
```

Any space surrounding these two operators is ignored.

Implementing `assert()`

Implementing `assert` reveals an important fact about using macros. Since the action of `assert` depends on the result of a test, you might first try an `if` statement, as in:

```
#define assert(cond) \
    if (!(cond)) __assert(#cond, __FILE__, __LINE__)
```

where the function `__assert` prints the message and halts the program. This implementation causes a problem, however, when `assert` finds itself within an `if` statement, as in:

```
if (x > 0)
    assert(x != y)
else
    /* whatever */
```

because the preceding code expands into

```
if (x > 0)
    if (!(x != y)) __assert("x != y", "file.c", 7);
else
    /* whatever */
```

The indentation that results from expanding `assert` in place is misleading because it's actually the second `if` that intercepts the `else`. Rewriting the expanded code to represent the actual flow of control produces:

```
if (x > 0)
    if (!(x != y))
        __assert("x != y", "file.c", 7)
    else /* OOPS! New control flow! */
        /* whatever */
```

The usual fix for nested `if` problems such as this is to use braces, as in:

```
#define assert(cond) \
{if (!(cond)) __assert
  (#cond, __FILE__, __LINE__)} }
```

Listing 4 Illustrates the token-pasting operator

```
/* trace2.c: Illustrate a trace *
 * macro for debugging */

#include <stdio.h>

#define trace(x,format) \
    printf("#x " = %" #format "\n",x)
#define trace2(i) trace(x ## i,d)

main()
{
    int x1 = 1, x2 = 2, x3 = 3;
    trace2(1);
    trace2(2);
    trace2(3);
    return 0;
}

/* Output:
x1 = 1
x2 = 2
x3 = 3
*/
/* End of File */
```

but this code expands into

```
if (x > 0)
    {if (!(x != y)) __assert
      ("x != y", "file.c", 7)};
else
    /* whatever */
```

and the combination `};` in the second line creates a null statement that completes the outer `if`, leaving a dangling `else`, which is a syntax error. A correct way to define `assert` is shown in Listing 5. (This simple version does not recognize the macro `NDEBUG`.) (Listing 6 shows the implementation of the support function `__assert()`). In general, when a macro must make a choice, it is good practice to write it as an expression and not as a statement.

Macro Magic

It's important to understand precisely what steps the preprocessor follows to expand macros, otherwise you can be in for some mysterious surprises. For example, if you insert the following line near the beginning of Listing 4:

```
#define x1 SURPRISE!
```

then `trace2(1)` expands into

```
trace(x ## 1,d)
```

which in turn becomes

```
trace(x1,d)
```

But the preprocessor doesn't stop there. It *rescans* the line to see if any other macros need expanding. The final state of the program text seen by the compiler is shown in Listing 7.

To further illustrate, consider the text in Listing 8. Listing 8 is not a complete program, by the way, but is for preprocessing only — don't try to compile it all the way. (If you have Borland C use the `CPP` command.) The output from the preprocessor appears in Listing 9. The `str()` macro just puts quotes around its argument. It might appear that `xstr()` is redundant, but there is an important difference between `xstr()` and `str()`. The output of the statement `str(VERSION)` is of course

Table 3 Preprocessor operators

Operator	Usage
#	Stringizing
##	Token pasting
defined	Symbol table query

Your Best CDROMs!



Cica MS Windows CDROM	4000 new Windows programs - quarterly updates	\$ 29.95*
Giga Games CDROM	3000 hot Games for MSDOS and Windows	\$ 39.95*
Space and Astronomy CDROM	Thousands of NASA images and data files	\$ 39.95*
C User Group Library CDROM	A collection of user supported C source code	\$ 49.95*
Simtel MSDOS CDROM	Classic: 650 MB Shareware / Freeware for MSDOS	\$ 29.95*
QRZ! Ham Radio CDROM	FCC Callsign Database for Win & DOS + files	\$ 29.95*
Gifs Galore CDROM	6000 mostly 640x480 GIF Images in every category	\$ 39.95*
Gutenberg Project CDROM	Classic Literature and historical documents	\$ 39.95
Hobbes OS/2 CDROM	600 MB always current Shareware/Freeware for OS/2	\$ 29.95*
Source Code CDROM	650 MB of Unix & DOS source code for programmers	\$ 39.95*
Internet Info CDROM	Thousands of computer, net, and internet documents	\$ 39.95
X11/GNU CDROM	X Windows and GNU software for Unix and SPARC	\$ 39.95
Aminet Amiga CDROM	650 MB new Shareware/Freeware for Amiga	\$ 29.95*
GEMini ATARI CDROM	650 MB 3000 Programs for Atari - also for BBS use.	\$ 39.95*
Ada CDROM	Programming tools, Ada source code and docs	\$ 39.95*
Nova for NeXT CDROM	600 MB black NeXT app's, src., docs, etc	\$ 59.95*
Nebula for NeXTSTEP Intel	600 MB NeXTSTEP Intel app's, docs, etc	\$ 59.95*
Garbo MSDOS/Mac CDROM	MSDOS and Macintosh Shareware/Freeware	\$ 29.95*
Fractal Frenzy CDROM	2000 high resolution fractals - royalty free	\$ 39.95
FreeBSD CDROM	Berkeley BSD. 32bit O/S for PC, w/ GNU & X11. Src	\$ 39.95
Linux CDROM	Yggdrasil Linux. 32bit O/S for PC, w/ GNU & X11. Src	\$ 49.95
La Coleccion CDROM	MSDOS, OS/2, Windows. Spanish descriptions	\$ 39.95*
Sprite CDROM	Berkeley's experimental Distributed OS for Sun's	\$ 29.95
CDROM Caddies	Standard type, highest quality, Lifetime guarantee!	\$ 4.95

1-800-786-9907
1-510-674-0783
FAX: 1-510-674-0821

email: orders@cdrom.com
Walnut Creek CDROM
4041 Pike Lane, Suite D-231
Concord, CA 94520

*Shareware programs
 require separate
 payment to authors if
 found useful
**All Our Discs Are
 Unconditionally
 Guaranteed.**

◆ Request 131 on Reader Service Card ◆



\$5995

for


\$69.95 for floppy
 disk distribution

32 Bit UNIX Compatible OS with:

X11R5, OPENLOOK, TCP/IP
 NFS, UUCP, C C++, Small Talk
 Objective-C, Pascal, lisp,
 f77, Perl System, V IPC, POSIX
 emacs, joe, vi, TeX, LaTeX, mail
 kermi, minicom, DOS emulation
 DOS partitions, SCSI, IDE, ESDI
 MFM, CDROM, VGA, S3, HGA

CGA, EGA, WD8993, WD8013
 NE2000, sound-blaster
 Complete documentation-
 full man pages included
 over 800 utilities
 386 with 4 meg required
 **/rdB** now available
 for Linux

Over 80,000 Users Worldwide



available

800-954-2829

LINUX SYSTEMS LABS.

18300 Tara Dr. • Clinton Twp., Michigan 48036
 (313) 954-2829 • FAX (313) 954-2806

GENERAL LINUX CONSULTING (United States only):
 1-900-446-6075, extension 835 ("TEK"), \$2.95/minute.
 10am-noon, 1:45pm-5pm Pacific Time.



◆ Request 158 on Reader Service Card ◆

The C Users Journal — March 1994

Page 107

"VERSION"

but *xstr(VERSION)* expands to

str(2)

because arguments not connected with a # or ## are *fully expanded* before they replace their respective parameters. The preprocessor then rescans the statement, providing "2". So in effect, *xstr()* is a version of *str()* that expands its argument before quoting it.

The same relationship exists between *glue()* and *xglue()*. The statement *glue(VERSION,3)* concatenates its arguments into the token *VERSION3*, but *xglue(VERSION,3)* first expands *VERSION*, producing

glue(2,3)

which in turn rescans into the token 23.

The next two statements are a little trickier:

```
glue(VERS,ION)
== VERS ## ION
== VERSION
== 2
```

and

```
xglue(VERS,ION)
== glue(VERS,ATILE)
== VERS ## ATILE
== VERSATILE
```

Of course, if *VERSATILE* were a defined macro it would be further expanded.

The last four statements in listing 8 expand as follows:

```
ID(VERSION)
== "This is version "xstr(2)
== "This is version "str(2)
== "This is version ""2"
```

```
INCFIL(VERSION)
== xstr(glue(version,2)) ".h"
== xstr(version2) ".h"
== "version2" ".h"
```

```
str(INCFIL(VERSION))
== #INCFIL(VERSION)
== "INCFIL(VERSION)"
```

```
xstr(INCFIL(VERSION))
== str("version2" ".h")
== #"version2" ".h"
== "\"version2\" \".h\""
```

The Portable C-database System

Available with CQL

herCules

■ Supports Database, Memo, and Index formats of dBase III, dBase IV, Clipper & FoxBase

■ Available with the CQL Implementation of ANSI Level 2 SQL

■ Ciphering and Password Mechanisms

SUPPORTS

UNIX SYS V, BSD, OS/2, Windows 3.X and NT, DOS, Novell NLN. Client/Server, ODBC, and library interfaces.

■ Royalty Free

■ Includes Shrouded Source

■ Compatible with any ANSI or K & R Compiler

■ Transaction Processing with Commit/Rollback

■ Documented Source Available

Machine Independent Software Corporation

491-B Carlisle Drive Herndon, VA 22070

703 435-0413 Fax: 437-8640 BBS: 437-8557

APIS Software

Bolongostraße 113 D-65929 Frankfurt, Germany

Country Code 49 Tel: [0]69 30 39 06

Fax: [0]69 31 75 31

◆ Request 180 on Reader Service Card ◆

Listing 5 A simple implementation of the assert macro

```
/* assert.h */

extern void __assert(char *, char *, long);

#undef assert
#ifdef NDEBUG
#define assert(cond)
    (void) 0
#else
#define assert(cond) \
    ((cond) \
     ? (void) 0 \
     : __assert(#cond, __FILE__, __LINE__))
/* End of File */
```

Listing 6 The __assert support function

```
/* xassert.c */
#include <stdio.h>
#include <stdlib.h>

void __assert(char *cond, char *fname, long lineno)
{
    fprintf(stderr,
        "Assertion failed: %s, file %s, line %ld\n",
        cond, fname, lineno);
    abort();
}

/* End of File */
```


For obvious reasons, the `#` operator effectively inserts escape characters before all embedded quotes and backslashes.

The macro replacement facilities of the preprocessor clearly offer you an incredible amount of flexibility (too much, some would say). There are two limitations to keep in mind:

- 1) If at any time the preprocessor encounters the current macro in its own replacement text, no matter how deeply nested in the process, the preprocessor does not expand it but leaves it as-is (otherwise the process would never terminate!). For example, given the definitions

```
#define F(f) f(args)
#define args a,b
```

F(g) expands to *g(a,b)*, but what does *F(F)* expand to? (Answer: *F(a,b)*).

- 2) If a fully-expanded statement resembles a preprocessor directive, (e.g., if expansion results in an `#include` directive), the directive is *not* invoked, but is left verbatim in the program text. (Thank goodness!).

Character Sets and Trigraphs

The character set you use to compose your program doesn't have to be the same as the one in which the program executes. These two character sets often differ in non-English applications. A C translator only understands the *source character set* — English alphanumeric, the graphics characters used for operators and punctuation (there are 29 of them), and a few control characters (newline, horizontal tab, vertical tab, and form-feed). Any other characters presented to the translator may appear only in quoted strings, character constants, header names or comments. The *execution character set* is the set of characters that the program uses in its literals, and to input and output data. This set is implementation-defined, but must at least contain characters representing alert (`'\a'`), backspace (`'\b'`), carriage return (`'\r'`), newline (`'\n'`), form feed (`'\f'`),

vertical tab (`'\v'`), and horizontal tab (`'\t'`).

Many non-U.S. environments use different graphics for some of the elements of the source character set, making it impossible to write readable C programs. To overcome this obstacle, standard C defines a number of *trigraphs*, which are triplets of characters from the Invariant Code Set (ISO 646-1983) found in virtually every environment in the world. Each trigraph corresponds to a character in the source character set which is not in ISO 646 (see Table 4). For example, whenever the preprocessor encounters the token `??=` anywhere in your source text (even in strings), it replaces this token with the `'#'` character code from the source character set.

COMPUTER LANGUAGE PRODUCT EXCELLENCE AWARD 1991

PC-lint for C/C++

presents Bug # 1729

```
#include <iostream.h>

class X
{
public:
    int upper;
    int lower;
    X(int init) : lower(init), upper(lower+1)
    {}
};

X x(1);
```

*This programmer expected the value of **lower** and **upper** of object **x** to be initialized to 1 and 2 respectively; instead, both were given the same value (1). How come? Call if you need a hint. Refer to Bug #1729.*

PC-lint for C/C++ will catch this and many other bugs. It will analyze a mixed suite of C and C++ modules to uncover bugs, glitches, quirks and inconsistencies.

Numerous C++ Warnings and Messages:

Are your inherited destructors virtual? Are your constructor `new`'s matched by your destructor `delete`'s? Are your initializers in order? Are names inadvertently hiding other names? Are your C++ modules consistent with your C modules? Much, much, more.

Plus Our Traditional C Warnings:

Uninitialized variables, unaccessed variables, possibly uninitialized variables, strong type mismatches, indentation irregularities, loss of precision, strange uses of Booleans, signed/unsigned mismatches, suspicious expressions, unused macros, etc. etc.

Full C++ Support - PC-lint for C/C++ is based on the ARM and is tracking the latest ANSI/ISO draft including exceptions and templates. It supports both Borland and Microsoft C/C++.

Options Galore: A plethora of options for message suppression, message format, compiler dependencies, etc. All messages can be individually suppressed or enabled, both locally and globally. Numerous compilers/libraries supported. Runs on MS-DOS (Optional built-in 386 DOS extender) and OS/2.

PC-lint for C \$139
PC-lint for C/C++ \$239

PC-lint users: call for update pricing
Unix and Mainframe programmers: call for pricing for FlexeLint.

Gimpel Software

3207 Hogarth Lane, Collegeville, PA 19426

CALL TODAY (215) 584-4261 Or FAX (215) 584-4266

30 Day Money-back Guarantee.

PA add 6% sales tax.

PC-lint and FlexeLint are trademarks of Gimpel Software

Listing 7 Preprocessed source with a surprise

```
main()
{
    int SURPRISE! = 1, x2 = 2, x3 = 3;
    printf("x1" " = %" "d" "\n", SURPRISE!);
    printf("x2" " = %" "d" "\n", x2);
    printf("x3" " = %" "d" "\n", x3);
    return 0;
}
/* End of File */
```

The program in Listing 11 shows how to write the "Hello, world!" program from Listing 10 using trigraphs. (Borland users: you have a separate executable, *trigraph.exe*, for processing trigraphs.)

In an effort to enable more readable programs world-wide, the C++ draft standard defines a set of digraphs and new keywords for non-ASCII developers (see Table 5). Listing 12 shows what "Hello, world" looks like using these new tokens. Perhaps you will agree that the symmetric look of the bracketing operators is easier on the eye.

Phases Of Translation

The C standard defines eight distinct phases of translation. An implementation doesn't make eight separate passes through the code, of course, but the result of translation must behave as if it had. The eight phases are:

1. Physical source characters are mapped into the source character set. This includes trigraph replacement and things like mapping a carriage return/line feed to a single newline character in MS-DOS.
2. All lines that end in a backslash are merged with their continuation line, and the backslash is deleted.
3. The source is parsed into preprocessing tokens and comments are replaced with a single space character. The C++ digraphs are recognized as tokens.
4. Preprocessing directives are invoked and macros are expanded. Steps 1 through 4 are repeated for any *included* files.

Heap Management for Windows and DOS

MoreHeap DOS/\$179, DOS & Windows/\$249

- Same API for Windows or DOS • Virtual memory (VM) for DOS and all MS-Windows modes • Swaps to extended, expanded memory and disk • VM binary trees (keyed and sorted access), multigigabyte arrays, stacks, queues, and linked lists for any data types—C++ class and C function interface • LRU algorithm ensures fast access to VM data • Emulate MS-Windows memory management API under DOS • Direct, linktime replacement for malloc function family • Just relink and malloc/new allocates from UMB, HMA, and EMM frame • Heap dump to video, file, or debug monitor with a single function call • Solves MS-Windows selector limitation problem • Solves heap fragmentation problems • Detects memory leaks, doubly freed pointers, and buffer overwrites • Works with SafeWin Windows debugger for the ultimate MS—Windows debug system • Complete source code included

SafeHeap \$95

- Link-time replacement for *str** and *mem** routines • Relink and run to detect buffer manipulation problems with globals, autos, and allocated blocks • Provides symbolic stack trace on error • Use with MoreHeap for enhanced debug capabilities • No memory overhead • Even debugs RTL and third party library function calls with no source code • Complete source code included

MegaHeap \$95

- A single, file i/o like interface to expanded/extended memory and disk • Automatically detects and uses resources • Just allocate a buffer and read and write to it like a file • Allocate up to 4GB • No block or transfer size limitations • Great for graphic images, large arrays, and more • Complete source code included

C-Shell \$125

- Replacement function for *exec* and *spawn* • Shrinks to a resident 5K kernel and spawns second application • Spawn a 600K program from a 600K program! • Swaps to expanded/extended memory and disk • Complete source code included

Call for demo disks.
Visa/MasterCard/COD/
POs accepted. 30 day
money-back guarantee.

Bundles
DOSPack (MoreHeap/DOS,
SafeHeap,MegaHeap, C-Shell)
\$299

WindowsPack (MoreHeap/
DOS & Windows, SafeWin)
\$399



Compatible with Borland C++
and Microsoft C++.

Seabreeze Software Systems
12 Tomlyn Drive
Princeton, NJ 08540

Sales/Support: 609-924-6793

BBS/FAX: 609-497-4607

e-mail/CIS: 72330, 705

5. Escape sequences in character constants and string literals that represent characters in the execution set are converted (e.g., '\a' would be converted to a byte value of 7 in an ASCII environment).
6. Adjacent string literals are concatenated.
7. Traditional compilation occurs: lexical and semantic analysis, and translation to assembly or machine code.
8. Linking occurs: external references are resolved and a program image is made ready for execution.

The preprocessor performs steps 1 through 4.

C++ And The Preprocessor

C++ preprocessing formally differs from that of C only in the tokens it recognizes. A C++ preprocessor must recognize the tokens in Table 5 as well as *.**, *->**, and *::*. It must also recognize *//*-style comments and replace them with a single space. Though

Listing 8 Illustrates macro rescanning

```
/* preproc.c: Test # and ## preprocessing operators
 *
 * NOTE: DO NOT COMPILE! Preprocess only!
 */

/* Handy stringizing macros */
#define str(s) #s
#define xstr(s) str(s)

/* Handy token-pasting macros */
#define glue(a,b) a##b
#define xglue(a,b) glue(a,b)

/* Some definitions */
#define ID(x) "This is version " #x xstr(x)
#define INCFILE(x) xstr(glue(version,x)) ".h"
#define VERSION 2
#define ION ATILE

/* Expand some macros */
str(VERSION)
xstr(VERSION)
glue(VERSION,3)
xglue(VERSION,3)
glue(VERS,ION)
xglue(VERS,ION)

/* Expand some more */
ID(VERSION)
INCFILE(VERSION)
str(INCFILE(VERSION))
xstr(INCFILE(VERSION))
/* End of File */
```

Listing 9 Preprocessed results from Listing 8

```
"VERSION"
"2"
VERSION3
23
2
VERSATILE

"This is version ""2"
"version2" ".h"
"INCFILE(VERSION)"
""\version2\ ""\ ".h""
```

◆ Request 283 on Reader Service Card ◆

C++'s preprocessor isn't much different than C's, you may want to *use* it a lot differently. For example, as far as I can tell, there is no good reason to define object-like macros anymore. You should use *const* variable definitions instead. The statement

```
const int MAXLINES = 500;
```

has a couple of advantages over

```
#define MAXLINES 500
```

Since the compiler knows the semantics of the object, you get stronger compile-time type checking. You can also reference *const* objects like any other with a symbolic debugger. Global *const* objects have internal linkage unless you explicitly declare them *extern*, so you can safely replace all your object-like macros with *const* definitions.

Function-like macros are *almost unnecessary* in C++. You can replace most function-like macros with inline functions. For

example, replace the *max* macro as shown previously with

```
inline int max(int x, int y)
{
    return x >= y ? x : y;
}
```

Note that you don't have to worry about parenthesizing to avoid precedence surprises, because this code defines a real function, with scope and type checking. You also don't have to worry about side effects like you do with macros, such as in the call

Table 4 Trigraph sequences

Trigraph	C Source Character
??=	#
??([
??/	\
??)]
??'	~
??<	{
??!	
??>	}
??-	~

Table 5 New C++ digraphs and identifiers

Token	Translation
<%	{
%>	}
<:	[
>:]
%%	#
bitand	&
and	&&
bitor	
or	
xor	^
compl	~
and_eq	&=
or_eq	=
xor_eq	^=
not	!
not_eq	!=

Energize Your DOS Applications with a Windows-Style Interface

Are your DOS applications starting to show signs of age? If so, it's time to do some remodeling. With our new **C/Windows Toolchest™**, you can easily create interfaces for your DOS applications that are similar to Microsoft® Windows™ applications.

Two simple function calls are all it takes to create movable, resizable **windows**, complete with **scroll bars** and other standard **controls**; including **minimize**, **maximize**, **restore**, and **menu buttons**. A wide variety of other controls are also available; including **push buttons**, **radio buttons**, and **check boxes**.

Of course there is extensive support for **menus**. Create horizontal and vertical menus, with or without scroll bars. Arrange menu items in a single row or column, or in multiple rows and columns. Attach a **sub-menu** to any menu item.

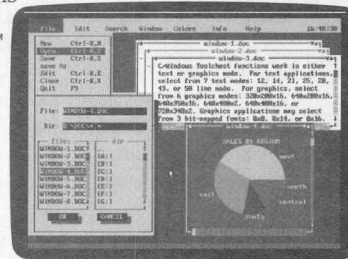
For collecting user input, you get a comprehensive set of functions to manage **data entry fields**. Collect data one field at a time, or all at once

through complete data entry forms. Use **picture clauses** to build data entry templates. Valid input may be enforced automatically, or you can define your own input validation functions.

Handling **mouse** input is easy. There are numerous high level mouse functions, including one that retrieves all mouse events and keystrokes. Mouse input is handled automatically by control buttons, menus, and data entry

fields. Low level mouse functions are also provided just in case you need them.

In all, the C/Windows Toolchest™ contains more than 250 functions to help you design a state-of-the-art user interface. Included are functions for implementing **context sensitive help**, **keyboard control**, and **graphics**. You also receive the complete source code for a multi-window **Notepad** editor that works in both text and graphics mode. C/Windows Toolchest™ works with C and C++ compilers from Mix®, Borland®, and Microsoft®.



Now One Interface Library Lets You Create Both Text and Graphics Mode Applications

<input type="checkbox"/> Please send FREE brochures	<input type="checkbox"/> C/Windows Toolchest.....\$39.95
Disk Size: <input type="checkbox"/> 5.25" <input type="checkbox"/> 3.5"	<input type="checkbox"/> Library Source.....\$10.00
(Requires DOS 2.0 or higher)	(Source code for C/Windows library)
Name _____	Shipping & Handling.....\$
Company _____	(\$5 USA, \$10 Canada, \$20 Foreign)
City _____	Texas Sales Tax (8.25%).....\$
State _____ Zip _____	Total Amount of Order.....\$
Country _____	Paying By: <input type="checkbox"/> Check or Money Order
Telephone _____	<input type="checkbox"/> Visa <input type="checkbox"/> MC <input type="checkbox"/> Amex <input type="checkbox"/> Discover
	Card# _____
	Exp. Date _____

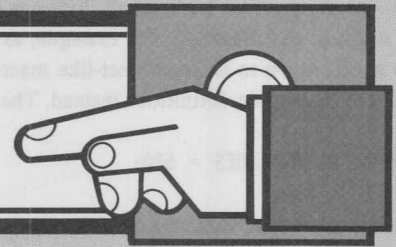


60 Day Money-Back Guarantee
7 x 9, 624 pages

To Order Please Call:
1-800-333-0330
Orders or Technical Questions:
Tel: 1-214-783-6001
Fax: 1-214-783-1404

Mix Software
1132 Commerce Dr.
Richardson,
TX 75081

Advertiser Index



Advertiser	Reader Service #	Pg. #	Advertiser	Reader Service #	Pg. #	Advertiser.....	Reader Service #	Pg. #
AccuSoft Corporation	140	24	FairCom	128	CIV	OPUS Software.....	159	87
ADONTEC GmbH.....	**	95	Fidelity Investments	**	23	Pacific Softworks	494	139
Advanced Design Solutions.....	242	92	Flanster Software.....	258	138	Paladin Software, Inc.....	227	100
AIB Software.....	150	58	FlashTek, Inc.....	276	64	Parsifal Software	300	141
AnSoft, Inc.....	307	141	Fortunel Systems, Inc.....	312	125	Powersoft Corporation	**	32
ARIS	135	143	General Software, Inc.	320	93	Professional Publications	317	139
Austin Code Works	313	62	General Software, Inc.	321	40	ProtoView Development Corp.....	376	15
AYECO, Inc.....	399	143	Gimpel Software	**	97	QNX Software Systems, Ltd.....	224	41
Az-Tech Software, Inc.....	245	139	Gimpel Software	**	109	Quadron	291	143
Been There - Done That Software	315	139	Gotoless Conversion	**	143	Quarterdeck Office Systems	201	13
Berard Software Engineering, Inc.	192	34	Green Hills Software, Inc.....	235	27	Raleigh Group International	193	45
Berard Software Engineering, Inc.	216	141	Greenleaf Software Inc.	240	CII	Ready-To-Run Software, Inc.....	155	25
Borland International.....	465	11	Ted Gruber Software	174	18	Realtime Control, Inc.....	191	138
Brightwork.....	269	143	Illustrated C	496	102	Rogue Wave Associates.....	124	9
Bristol Technology, Inc.....	309	42	ImageSoft, Inc.....	177	139	Ryle Design	110	74
Bristol Technology, Inc.....	259	16	IMCSI.....	170	139	Scientific Endeavors	172	12
Burton Systems Software	430	31	Information Modes.....	118	66	SeaBreeze Software Systems.....	283	110
Byte Dynamics, Inc.	125	144	Innovative Software	223	39	Sequiter Software, Inc.....	257	CIII
Bytech Business Systems	229	59	InstantInfo Index	**	135	SET Laboratories, Inc.	210	86
C Associates	137	140	Intelligent Solutions Inc.	184	141	Sigma Software, Inc.....	114	144
CAD-UL GmbH	209	96	Intelligent Systems Int'l	206	88	S.I.P. - Software Solutions	230	36
Catenary Systems	253	90	Interactive Instruments	120	116	SLR Systems, Inc.....	268	73
C_Graph Software, Inc.....	261	144	International Computer Prof. Assoc. ...	108	140	SofDesign International.....	263	61
ChipTools, Inc.....	**	138	Intertech, Inc.....	318	129	SofDesign International.....	215	68
Chirp Technical Services.....	**	142	InTrepid Software.....	270	123	Softronic GmbH	225	17
CMX Company.....	325	144	ITI Logiciel	**	141	Software Blacksmiths, Inc.....	147	103
Cobalt Blue, Inc.	105	37	Just Logic Technologies, Inc.	295	99	Software Blacksmiths, Inc.....	142	143
Computer Language Arts, Inc.	440	98	KADAK Products Ltd.....	**	43	Software Frontiers	282	142
Courseware Applications, Inc.....	130	143	Linux Systems Laboratories	158	107	The Software Group	190	75
Creative Programming.....	123	63	Little Wing	119	130	Software Interphase	220	22
Crystal Software, Inc.....	**	53	Machine Independent Software	180	108	Software Science, Inc.	324	51
CUG CDROMs.....	251	124	Mark Williams Company	**	1	Softway, Inc.....	186	8
CUG Directory IV	277	60	MarshallSoft Computing, Inc.....	112	140	Spider Software.....	323	143
C User's Bookstore.....	153	81	µC/OS	127	79	Strategic Software Designs Inc.....	387	38
Datalight.....	256	84	Micro Digital, Inc.	202	114	StratosWare Corporation.....	284	7
DBx, Inc.....	207	54	Micro-Processor Services, Inc.	162	141	Sub Systems, Inc.....	452	69
DC Micro Development.....	152	138	MicroQuill Software Publishing.....	**	140	The Symmetry Group	316	85
DCW Group.....	**	138	Microsoft Corp.....	**	2	T & T Computer Products.....	247	52
DDC International.....	218	14	Microware Sas	204	49	Technosoft.....	250	57
Diab Data, Inc.....	154	20	MIX Software	236	105	Tics Realtime	308	30
Digital Computing System.....	122	139	MIX Software	146	111	Trio Systems	**	67
D&L OnLine, Inc.....	301	143	MMC AD Systems.....	134	141	Valois Software	287	144
DMARZ Diversified Sciences Co.....	310	139	Network Dynamics	132	139	Voice Information Systems, Inc.	238	141
Eagle Software, Inc.....	183	72	Nu-Mega Technologies.....	**	16a	Walnut Creek CDROM.....	131	107
E-Data Link.....	181	142	Object Management Laboratory	106	29	Willies' Computer Software Co.	304	55
EliaShim Microcomputers, Inc.....	144	53	Object Oriented Software Engineering	196	116	Windbase Software.....	178	89
Ematek GmbH.....	219	28	Omega Point, Inc.....	136	57	Windows Custom Controls.....	278	128
Embedded System Products, Inc.....	380	127	On Line Magazine Index	**	78	Windows/DOS Developer's Journal.....	**	121
Emerald Bay Group	296	104	On Time Marketing	214	56	XVT Corporation.....	197	19
Emerging Technology	111	113	Opt-Tech Data Processing	203	141	Zinc Software	126	6
EMS.....	281	142						

This index is provided as a service to our readers. The publisher assumes no liability for errors or omissions.

***This advertiser prefers to be contacted directly.*

```
max(x++,y++)
```

The macro version may seem superior to the inline function because it accepts arguments of any type. No problem. Define *max* as a template, as in the following code; now the inline function will accept arguments of any type:

```
template<class T>
inline int max(const T& x, const T& y)
{
```

```
    return x > y ? x : y;
}
```

Do keep in mind, however, that *inline* is a only *hint* to the compiler. Not all functions are amenable to inlining, especially those with loops and complicated control structures. Your compiler may tell you that it can't inline a function. Still, in many cases it is better to define a function out-of-line than to define it as a macro and lose the type safety that a real function affords.

There is still room in C++ for function-like macros that use the stringizing or token-pasting operators. The program in Listing 13 uses stringizing and an inline function to test the new string class available with Borland C++ 4.0.

Listing 10 A "Hello, world!" program

```
/* hello.c: Greet either the user or the world */
#include <stdio.h>

main(int argc, char *argv[])
{
    if (argc > 1 && argv[1] != NULL)
        printf("Hello, %s!\n", argv[1]);
    else
        printf("Hello, world!\n");
    return 0;
}

/* End of File */
```

Listing 11 "Hello, World!" using trigraphs

```
/* thello.c: Greeting program using trigraphs */
#include <stdio.h>

main(int argc, char *argv??(??))
??<
    if (argc > 1 && argv??(0??) != NULL)
        printf("Hello, %s!??/n", argv??(1??));
    else
        printf("Hello, world!??/n");
    return 0;
??>

/* End of File */
```

Listing 12 "Hello, World!" with the new C++ digraphs and tokens

```
/* dhello.c: Greeting program using C++ digraphs */
#include <stdio.h>

main(int argc, char *argv<:>)
<%
    if (argc > 1 and argv<0:> != NULL)
        printf("Hello, %s!<?/?/n", argv<1:>);
    else
        printf("Hello, world!<?/?/n");
    return 0;
%>

/* End of File */
```

Conclusion

The preprocessor doesn't know C or C++. It is a language all its own. Many library vendors have used the preprocessor intelligently to simplify the installation and use of their products. I encourage you to use it, but to use it prudently. It has some dark corners, which I've purposely avoided. It is good practice, especially with C++, to do as much as you can in the programming language, and use the preprocessor only when you need to. □

EDIX™

Professional Text Editor

Since 1982, EDIX has been the text editor of choice for professional programmers. EDIX's no-nonsense human interface, along with its speed, power and flexibility make it an invaluable productivity tool.

- Compatible versions for Windows, DOS, OS/2, and UNIX (SCO, HPUX, AIX, SunOS, and others).
- User configurable.
- Execute any program from within EDIX.
- Powerful block handling features, including column handling.
- Powerful search and replace.
- On-line help and tutorials.

For more information, please call 303-447-9495 or send us a fax at 303-447-9241. Demos are available at no charge.

Emerging Technology
2888 Bluff Street, Suite 263
Boulder, CO 80301

◆ Request 111 on Reader Service Card ◆

```
// tstr.cpp:    Test the C++ string class

#include <iostream.h>
#include <stddef.h>
#include <cstring.h>

// Handy display macros
#define result(exp) \
    cout << #exp ":  \"" << (exp) << "\" << endl
#define test(obj,exp) \
    exp, print(#obj, after " #exp ":\n",obj)

// Print a string in quotes
inline void print(const char *p, const string& s)
{
    cout << p << " " << s << " " << endl;
}

main()
{
    string s1("Now is the time for all worthy carbon units"),
        s2 = "to come to the aid of their sector.",
        s3 = '\n',
        s4(s1);
    size_t len = s1.length();
```

```
// Test some operators
result(s1 == s4);
result(s1 < s4);
result(s1 + s3 + s2);
test(s1,s1 += s3 + s2);
result(s1 == s4);
test(s1,s1.resize(len));
result(s1 == s4);
cout << endl;
```

```
// Search and replace
size_t pos = s1.find("all");
if (pos != NPOS)
    test(s1,s1.replace(pos,3,"some"));
pos = s1.find("worthy");
if (pos != NPOS)
{
    result(s1.substr(pos,5));
    test(s1,s1.insert(pos,"un"));
}
cout << endl;
```

```
// More searching
result(s1.find_first_of("aeiou"));
result(s1.find_first_not_of("aeiou"));
result(s1.find_last_of("aeiou"));
result(s1.find_last_not_of("aeiou"));
cout << endl;
```

```
// Subscripting
pos = s2.find_first_of('d');
test(s2,s2[pos] = 'l');
return 0 ;
}
```

```
/* Output:
s1 == s4:  "1"
s1 < s4:  "0"
s1 + s3 + s2:  "Now is the time for all worthy carbon units
to come to the aid of their sector."
s1, after s1 += s3 + s2:
"Now is the time for all worthy carbon units
to come to the aid of their sector."
s1 == s4:  "0"
s1, after s1.resize(len):
"Now is the time for all worthy carbon units"
s1 == s4:  "1"
```

```
s1, after s1.replace(pos,3,"some"):
"Now is the time for some worthy carbon units"
s1.substr(pos,5):  "worth"
s1, after s1.insert(pos,"un"):
"Now is the time for some unworthy carbon units"
```

```
s1.find_first_of("aeiou"):  "1"
s1.find_first_not_of("aeiou"):  "0"
s1.find_last_of("aeiou"):  "43"
s1.find_last_not_of("aeiou"):  "45"
```

```
s2, after s2[pos] = 'l':
"to come to the ail of their sector."
*/
```

```
// End of File
```


v3.1

SIMPLE MULTITASKING EXECUTIVE

BEST 80x86 KERNEL

- task manager
- memory manager
- intertask comm.
- error manager
- i/o, events, & timers
- resource manager
- preemptive
- ROM'able
- fast and small
- no royalty

EASY TO USE

- libraries for Microsoft C/C++, Borland C/C++, High C/C++, Symantec C++, and Phar Lap
- Quick Start, User's Guide, and Ref. manuals
- load & go platforms for all products
- 6 months free support and updates

ADVANCED FEATURES

- 16-bit protected mode
- 32-bit protected mode (flat or segmented)
- C++ class library (**smx++**)
- DOS-compatible file manager (**smxFile**)
- Dynamic Load Module support (**smxDLM**)
- task-level debugger (**smxProbe**)

✓ TCP/IP support (smxNet) NEW!

MICRO DIGITAL
 6402 Tulagi St.
 Cypress CA 90630-5630

Call for Information:
1-800-366-2491
 FAX 714-891-2363

◆ Request 202 on Reader Service Card ◆

Title: *C Elements of Style
The Programmer's
Style Manual
for Elegant C and
C++ Programs*
Author: Steve Oualline
Publisher: M&T Books
411 Borel Ave, Suite 100
San Mateo, CA 94402
1992
Price: \$21.95
Pages: 265
ISBN: 1-55851-291-8

C Elements of Style

reviewed by Dwayne Phillips

Steve Oualline's *C Elements of Style* is, as its name suggests, a style manual for C programming. Oualline outlines a method for writing clear and decipherable C code. He emphasizes a simple and straightforward style. In his words, this book is for programmers "who want their programs to be easily read and maintained by others."

Audience

You do not need to be a C wizard to understand this book. In fact, wizards who write statements like:

```
*destination++ = *source++
```

may not like this book. Oualline advocates the more accessible:

```
*destination = *source; destination++; source++;
```

If you write programs that must be corrected or augmented by others, this book will be of interest to you. It is especially appropriate for relatively new programmers who are struggling to learn good C programming habits.

Contents

C Elements of Style has nine chapters, a compact style manual, two appendices, and an index. The chapters cover the topics expected in this type of book. They include (1) style and program organization, (2) file basics, comments, and program headings, (3) variable names, (4) statement formatting, (5) statement details, (6) the preprocessor, (7) C++ style, (8) directory organization and makefile style, and (9) user-friendly programming.

For much of the book, Oualline gives examples of good and bad code, and summarizes with style rules. His rules are simple and easy to apply. Some of the rules apply to the process of writing code. For example, "Comment your code as you write it." (This rule, he says, saves time in the long run.) Other rules address the code itself: "Constant names are all upper case" and "Follow every variable declaration with a comment that defines it." Some rules just make a programmer's life easier: "Assume that *, /, and % come before + and -. Put parentheses around everything else."

The style rules are aimed at making programs readable and reliable. Good variable and subroutine names, which help make the code explain itself, and good comments, which help reveal the organization of programs, enhance readability. Oualline strives to increase reliability by limiting the code to safe subsets of the C language. (Not allowing shortcuts such as `*destination++ = *source++` reduces the risk of problematic side effects.) Oualline describes this emphasis on readable and reliable code as "defensive programming" or "doing a lot of thinking so I don't have to do a lot of thinking."

The chapter on makefile style is outstanding and to my knowledge unique. Makefiles are essential to C programming and are often the hardest part of a project to understand. Other books describe how to use makefiles, but this is the only book I have seen that tells how to organize and format them. Oualline formally defines many makefile items that have become pseudo standards over the years. This material may seem trivial to an experienced UNIX C programmer, but many C programmers today have never worked on a UNIX system (believe it or not).

The chapter on user-friendly programming is a bit out of place, but useful. It discusses how to write programs that users will actually use. (It includes the Law of Least Astonishment: The program should act in a way that least astonishes the user.)

Dwayne Phillips works as a computer and electronics engineer with the U.S. Department of Defense. He has a PhD in Electrical and Computer Engineering from Louisiana State University. His interests include computer vision, artificial intelligence, software engineering, and programming languages.

The style manual is 35 pages of rules without all the discussion. For those short on time, this book-within-a-book provides a concise summary of the text.

The first appendix shows three complete code examples (two in C, one in C++) that employ the style rules given in the body of the book. These examples show Oualline's recommendations in practice. The second appendix lists all the rules given in the chapters. (Take these seven pages and pin them on the wall next to the coffee machine.)

Other Style Manuals

For years, the only programming style manual available was Kernighan and Plauger's thin little classic *The Elements of Programming Style* [1]. Fortunately, several style manuals have appeared in the recent past — each written with a different point of view.

Plum's *C Programming Guidelines* [2] and *C++ Programming Guidelines* [3] (reviewed in *The C Users Journal*, January 1993) are comprehensive references on programming standards and style. They are written in a compact, subroutine-like style that is appropriate for reference books. These books are technical and not for the novice.

Steve McConnell's *Code Complete* [4] is a comprehensive handbook for programmers. As such, it includes plenty of good advice on programming style, but its style sections are scattered throughout the 800-plus pages.

While Oualline favors code that anyone can understand, Ranade and Nash's *The Elements of C Programming Style* [5] (reviewed

in *The C Users Journal*, July 1993) encourages programmers to master and exploit the notation of the C language. (Ranade and Nash's do recommend backing away from terse C style if the audience includes programmers who have a limited knowledge of C.)

Conclusion

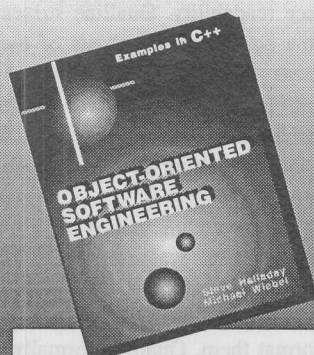
Oualline's book is the closest to Kernighan and Plauger's and it serves as a worthy update to that earlier classic. It is a short, yet complete treatment of C style.

C Elements of Style is a deserving addition to the desk top — not just the bookshelf. □

References

- [1] *The Elements of Programming Style*, second edition, Brian W. Kernighan, P.J. Plauger, McGraw Hill, New York, New York, 1978, ISBN 0-07-034207-5.
- [2] *C Programming Guidelines*, second edition, Thomas Plum, Plum Hall Inc., ISBN 0-911537-07-4.
- [3] *C++ Programming Guidelines*, Thomas Plum and Dan Saks, Plum Hall Inc., ISBN 0-911537-10-4.
- [4] *Code Complete, A Practical Handbook of Software Construction*, Steve McConnell, Microsoft Press, One Microsoft Way, Redmond, Wash. 98052-6399, ISBN 1-55615-484-4.
- [5] *The Elements of C Programming Style*, Jay Ranade and Alan Nash, McGraw Hill, New York, New York, 1993, ISBN 0-07-051278-7.

IMPROVE YOUR SOFTWARE ENGINEERING WITH OBJECT-ORIENTED METHODS



OBJECT-ORIENTED SOFTWARE ENGINEERING

by
Steve Halladay &
Michael Wiebel

Learn how to:

- Engineer applications to minimize costs
- Manage all six phases of the software lifecycle
- Use recursion in the object-creation process

USE RECURSION FOR PRECISE DESIGN MANAGEMENT

This book demonstrates object-oriented principles for each phase of development, from specification through maintenance. A comprehensive example coded in C++ ties the demonstration together. For precision, the design method is presented in a pseudocode recursive algorithm.

ONLY
\$29.95
Plus
Shipping

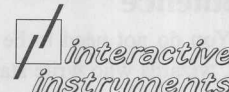
ORDER TODAY!

Order: Book T37 *Object-Oriented Software Engineering*

913-841-1631
FAX 913-841-2624



For software architects ...



- ☐ thinking about friendly end-user programming tools
- ☐ drafting flexible text manipulators
- ☐ in need to organize rapid access to huge amounts of data

... we have the building blocks

TextMatch

Table driven text converter plus general purpose macro processor

KeyPoint

Balanced binary trees, abbreviated keywords associator, constructor for list-directed languages

DataOrgan

Sequential and keyed record set manager with B*-trees, based on time and space efficient page oriented data storage

All written in Standard C, easily fit into C++.

For complete capability, platform and price profile inquire

interactive instruments

Software for Science and Engineering

Beethovenplatz 14
53115 Bonn · Germany
CompuServe: 73064,1240

Phone: +49-228-650041
Fax: +49-228-697608
MasterCard accepted

IOCCC, ASXXXX, MINED, TDE Update, and a Bug Fix

CUG Library Volume IV

CUG Library proudly announces volume IV of its directory of user-supported C source code. This latest effort thoroughly catalogs CUG Library volume numbers #300 through #349. Volume IV includes both exhaustive reviews of selected volumes plus capsule summaries of all volumes. In all, this amounts to more than 250 cross-referenced and indexed pages of information. Volume IV can be yours for \$10 or order the set of volumes I through IV for just \$28 total. As always, see the order blank in the center portion of this issue.

Bug Fix for GNUPlot, CUG #334

Arild Olsen <d_olsen_o@kari.uio.no> writes:

"I just received the disk, and the HPLJII-driver does not function, as stated by R.T. Stevens in his review (CUJ, June 93). Maybe this is due to an incompatibility between HP LaserJet II and III; I have a LJ III. When sending a bitmap to the printer, the program specifies TIFF-format. This is not correct since the bitmap is plain.

To solve this, edit the *HPLJIItext* function in the HPLJII driver. The string "`\033*b2m%dW`" should be changed to "`\033*b0m%dW`".

New Acquisitions and Updates

This month we present three additions to the CUG Library, as well as an update to a recently featured volume.

- International Obfuscated C Code Contest (CUG #397)
- ASxxxx Cross Assembler - Part 3 (CUG #398): MC 68HC08 CPU support
- MINED Editor: (CUG #399)

- Thomson-Davis Editor (CUG #386 update): multi-window text/binary file editor — new version 3.2A

International Obfuscated C Code Contest 1984-1993: CUG #397

Landon Noll (Sunnyvale, CA) submits a decade of source code from the International Obfuscated C Code Contest (IOCCC). This contest has long been a favorite of many *CUJ* readers. The entire IOCCC archive from 1984-1993 is now available as a two-diskette set from the CUG Library. Obfuscation implies purposefully obscuring and confusing a situation. Why obfuscate C code? The official IOCCC states its objectives as follows:

- To show the importance of programming style, in an ironic way.
- To stress C compilers by feeding them unusual code.
- To illustrate some of the subtleties of the C language.
- To provide a safe forum for poor C code.

Recently, Bob van der Poel reviewed Don Libes' book entitled *Obfuscated C and Other Mysteries* (see *CUJ*, October 1993, pp. 131-132). The diskette for this book includes IOCCC entries from 1984-1991. Libes has produced special reports about the IOCCC several times

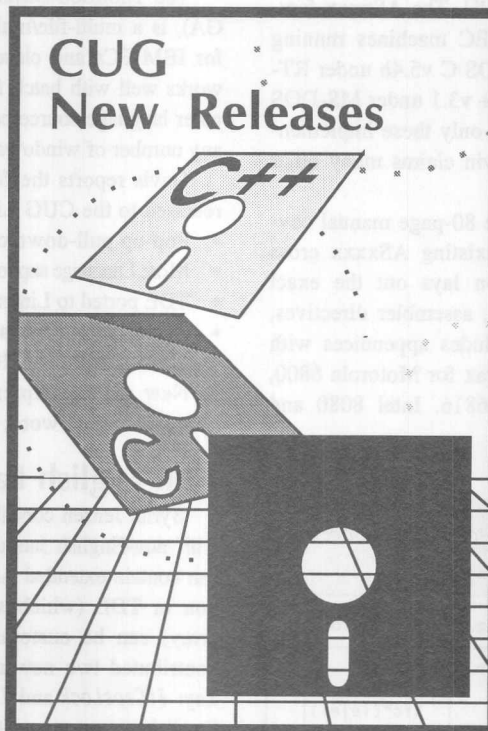
in *CUJ*. Please see the following back issues for more detail:

Libes, Don. "Don't Put This on Your Resume," *CUJ*, May 1991, p. 89.

Libes, Don. "The Far Side of C," *CUJ*, May 1990, p. 125.

Libes, Don. "The International Obfuscated C Code Contest," *CUJ*, July 1989, p. 93.

The CUG Library volume #397 contains the full IOCCC archive including two additional years not included in the Libes' book.



Victor R. Volkman received a BS in Computer Science from Michigan Technological University. He has been a frequent contributor to The C Users Journal since 1987. He is currently employed as Senior Analyst at H.C.I.A. of Ann Arbor, Michigan. He can be reached by dial-in at the HAL 9000 BBS (313) 663-4173 or by Usenet mail to sysop@hal9k.com.

In addition to dozens and dozens of obfuscated C programs, the archive includes complete rules and guidelines so you can submit your own entries into next year's contest. Some of the obfuscated programs are quite useful, including scaled-down versions of make, grep, and various editors.

ASxxxx Cross Assembler - Part 3: CUG 398

Cross assemblers continue to play an important role in the CUG Library. A cross assembler reads assembly language source code for a non-native CPU and writes object code that can be linked and downloaded to the target machine for execution. Embedded systems developers are the most frequent users of cross assemblers. This month, Alan R. Baldwin (Kent State University, Ohio), adds his third cross assembler to the CUG Library's repertoire. ASxxxx Part 3 provides a complete Motorola 68HC08 development system. ASxxxx Part 3 version 1.50 (released 8/9/93) is immediately available as CUG volume #398.

The CUG distribution of ASxxx Part 3 includes MS-DOS executables for the ASxxxx Cross Assembler and Linker. However, if you want to recompile the Cross Assembler or Linker, you'll also need ASxxxx Part 1 (CUG #292). ASxxx Part 2 contains cross assembler source files for the 6816 CPU. The ASxxxx family of cross assemblers can be built on DEC machines running DECUS C in the TSX+ environment or PDOS C v5.4b under RT-11. ASxxxx has been built with Borland C++ v3.1 under MS-DOS and includes a project (.PRJ) file. Although only these implementations have been specifically tested, Baldwin claims many other K&R C compilers may work as well.

ASxxxx Part 3 includes a comprehensive 80-page manual covering functionality provided by all three existing ASxxxx cross assemblers and linkers. The documentation lays out the exact specifications of syntax for symbols, labels, assembler directives, and expressions in detail. The manual includes appendices with instruction set highlights and supported syntax for Motorola 6800, 6801, 6804, 6805, 68HC08, 6809, 6811 6816, Intel 8080 and 8085, and Zilog Z80 and HD64180 CPUs.

The ASxxxx assembler falls short of full macro implementation, but does include a host of important features such as: if/then/else, #include files, radix support from binary to hexadecimal, and a full complement of C-language operators for expressions. The ASxxxx linker goes beyond conventional loaders by resolving intermodule symbols, combining code into segments, relocating absolute symbols and base addresses, and producing either Intel HEX or Motorola S19 output files.

MINED Editor: CUG #399

MINED, by Thomas Wolff (Freie Universität Berlin, Institut für Informatik, Germany), is a modeless full-screen text editor. MINED was originally written for MINIX and now works with most UNIX platforms as well as MS-DOS, and DEC VAX-11/VMS. MINED works best at editing small files (50K or less) and can edit many files simultaneously. Unlike other editors which have separate command modes and input modes, MINED uses a modeless design for ease of use. It also includes powerful regular expression operations for both searching and replacing text. MINED Version 3 (released 08/04/93) is immediately available as CUG volume #399.

Thompson-Davis Editor Update: CUG #386

The Thomson-Davis Editor, as provided by Frank Davis (Tifton, GA), is a multi-file/multi-window binary and text file editor written for IBM PCs and close compatibles. Thomson-Davis Editor (TDE) works well with batch files, binary files, text files, and various computer language source code files. TDE can handle any size of file and any number of windows that fit in conventional DOS memory.

Davis reports the following enhancements since TDE was last released to the CUG Library:

- Pop-up pull-down command menu = <CTRL>+)
- More Language support, thanks to Byrial Jensen, <byrial@daimi.aau.dk>
- TDE ported to Linux (POSIX, SVR4, BSD4.3+?, FIPS 151-1, etc.)
- A bug (a blunder, actually) got fixed in the 3.1 config utility.
- Linux FAQs and HOWTOs
- New regular expression meta characters: < = Empty string at beginning of word; > = Empty string at end of word

Non-English Language Support

Byrial Jensen contributed several functions to TDE that are useful with non-English languages. Using these functions, DOS filenames can contain extended ASCII characters. As a result, the *dirlist* function in TDE (which sorts filenames according to the sort order array) can be customized to your favorite alphabet. Byrial also contributed two new macro functions that look at the Caps Lock key: *IfCapsLock* and *IfNotCapLock*. Other changes supporting non-English usage are as follows: predefined regular expression macros may be redefined; all editor prompts have been gathered into *prompts.h*; response letters have been gathered into *letters.h*; and the window letters can be changed to follow a non-English alphabet.

Improved Regular Expression Handling

Davis writes: "I use the regular expression search much more often than I first anticipated. A couple of features missing in the original implementation are the beginning-of-word and end-of-word metacharacters. These metacharacters really come in handy for culling prefixes and suffixes from the search. Here's our new regular expression table: [Please refer to Table 1]"

TDE version 3.2a (Released 11/15/93) immediately replaces version 3.0 and is available as CUG Library volume #386. □

Table 1 Regular Expression operator precedence:
(based on the table in Dr. Aho's book)

c = char x = string r,s = regular expression		
c	any non-operator character	Felis
\c	c literally and C escapes	catus\.
\:c	predefined macro:	\:c*(ie ei)
	\:a - alphanumeric	
	\:b - white space	
	\:c - alphabetic	
	\:d - decimal	
	\:h - hex	
	\:l - lower alpha	
	\:u - upper alpha	
.	any character but newline	c.t
<	beginning of word	<cat
>	end of word	<cat>
^	beginning of line	^cat
\$	end of line	cat\$
[x]	any character in x	[a-z0-9]
[^x]	any character not in x	[^AEIOU]
r*	zero or more r's	ca*t
r+	one or more r's	ca[b-t]+
r?	zero or one r	c.?t
rs	r followed by s	^s
r s	either r or s	kitty cat
(r)	r	(c)?(a+t)

C++ SIM

Introduction

This month's product focus is derived from documentation written by M.C. Little and D.L. McCue. Little and McCue have provided this documentation, which describes their C++ SIM simulation class library, expressly for reprint in *The C Users Journal*.

C++ SIM is a class library which provides discrete process-based simulation similar to that provided by SIMULA [Birtwhistle 73][Dahl 70] and has been used in the work presented in [McCue 92]. Based on the facilities provided in SIMULA, C++ SIM provides active objects (instances of C++ classes) as the units of simulation using the type-inheritance facilities of C++ to convey the notion of "activity."

C++ SIM is designed to be used with a user-supplied threads package. C++ SIM's authors use Sun Microsystems's lightweight process (thread) package; however, they have added this package to the simulation class hierarchy through an abstract class definition so that other lightweight process packages can be used instead with very little modification. Users of this framework can replace existing classes as long as the replacements conform [Black 86] to the original class definition.

This article describes the C++ SIM class hierarchy, and shows how it can be used to further refine the simulation package [1].

The Class Hierarchy

Figure 2 illustrates the main class hierarchy within the simulation package. The base class is *Thread*, which provides the mini-

mum functionality required of a threads library. Two classes, *GNU_Thread* and *LWP_Thread*, derive from the *Thread* class to support the threads packages that were available to C++ SIM's authors at the time of this writing. These classes are Sun's own lightweight process package, and the GNU threads library. Class *Thread_Type*

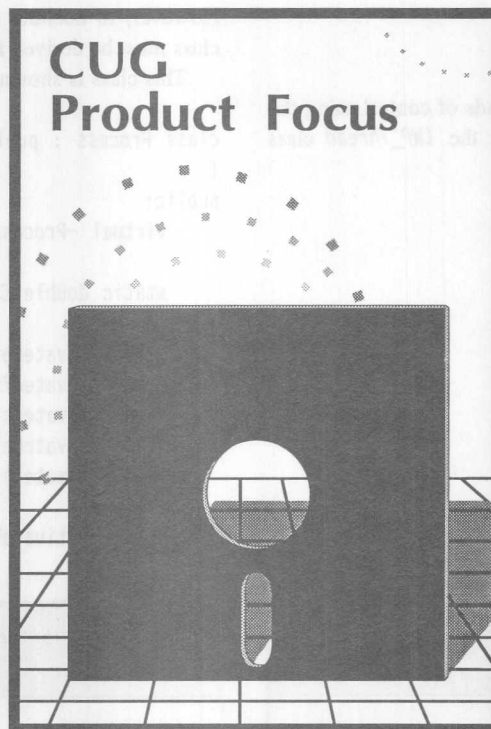
provides a (relatively) transparent way to change from one thread implementation to another. Class *Process* provides all operations required by the simulator to control execution of all processes in the simulation. These classes are described in the following sections.

The Threads Base Class

In keeping with the C++ programming model, classes obtain the thread characteristic, necessary to convey the notion of "activity" within the simulation environment, by inheriting an appropriate base class (in simulation terms they become processes). In C++ SIM, all classes that provide the abstraction of threads must be derived from the *Thread* base class. This base class forces the derived class to provide a minimum set of operations required for the management of threads. (The base class defines these operations as pure virtual functions, and C++ requires that a deriving class define such functions before an instance of the class can be declared.) These operations

are shown in the *Thread* class as follows:

```
class Thread
{
public:
    virtual void Suspend() = 0; // pure virtual function
    virtual void Resume() = 0;
```



Victor R. Volkman received a BS in Computer Science from Michigan Technological University. He has been a frequent contributor to *The C Users Journal* since 1987. He is currently employed as Senior Analyst at H.C.I.A. of Ann Arbor, Michigan. He can be reached by dial-in at the HAL 9000 BBS (313) 663-4173 or by Usenet mail to sysop@ha19k.com.

```

virtual long Identity();
static Thread* Self();
};

```

When defined, the *Suspend* and *Resume* methods will give the thread package specific ways of suspending and resuming execution of a thread respectively.

Body represents the controlling code for each object, i.e., the scope within which the controlling thread will execute.

Current_Thread must be defined by the derived class, since it returns the identity of the currently executing thread, which is specific to the thread package used.

The base class itself implements the operations *Identity* and *Self* because some threads packages do not provide similar functionality. *Identity* returns the unique identity of the thread associated with a given object, and *Self* returns the currently executing thread. Because *Self* is a static member function programs can invoke it without creating an instance of the Thread class, i.e., programs may call *Thread::Self()*.

The Class LWP_Thread

User classes which require separate threads of control using the Sun thread package can be derived from the *LWP_Thread* class shown as follows:

```

class LWP_Thread : public Thread
{
public:
    virtual void Suspend();
    virtual void Resume();
    virtual void Body() = 0;

    virtual long Current_Thread();

    thread_t Thread_ID();

```

```

protected:
    static const int MaxPriority;
    LWP_Thread(int priority = MaxPriority);
};

```

The *MaxPriority* constant represents the maximum priority at which a thread may execute (by default all threads derived from this class execute at this priority). Class *LWP_Thread* defines all of the pure virtual functions declared in *Thread* except *Body*, which must be defined by the deriving class.

Initialize initializes the threads package prior to use. (Obviously the operations performed within this method are thread package specific.)

Thread_ID returns more detailed (package-specific) information about the associated thread.

The Process Class

Applications could derive from the *Thread* base class to provide active objects in C++ outside of the simulation package. However, to become a process in the simulation environment, a class must be derived from the *Process* base class.

This class is shown as follows:

```

class Process : public LWP_Thread
{
public:
    virtual ~Process ();

    static double CurrentTime ();

    void ActivateBefore (Process&);
    void ActivateAfter (Process&);
    void ActivateAt (double AtTime = CurrentTime());
    void ActivateDelay (double AtTime = CurrentTime());
    void Activate();

    void ReActivateBefore (Process&);

```

Figure 1 Head of simulation queue

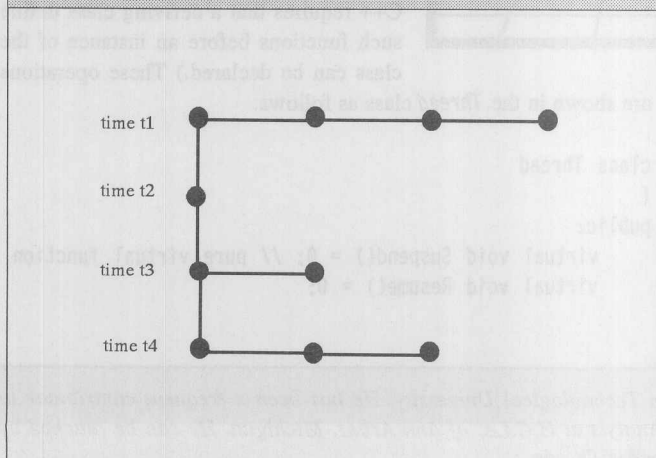
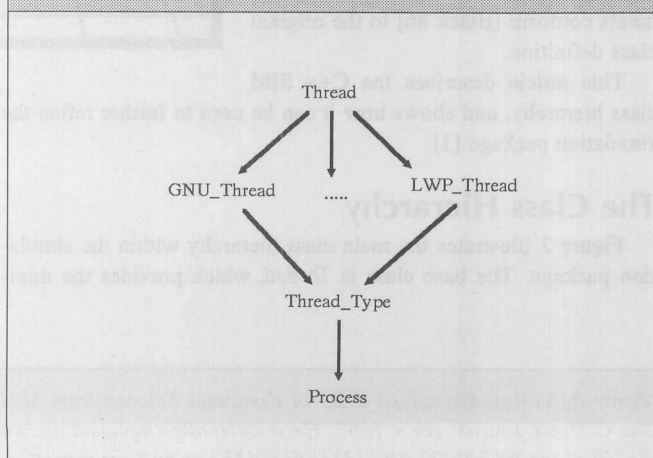


Figure 2 Simulation class hierarchy




```

void ReActivateAfter (Process&);
void ReActivateAt (double AtTime = CurrentTime());
void ReActivateDelay (double AtTime = CurrentTime());
void ReActivate ();

void Cancel ();
double evtime ();
void set_evtime (double);

boolean idle ();
boolean terminated ();

virtual void Body () = 0;

protected:
    Process ();

    void Hold (double t);
    void Passivate ();
};

```

idle returns either TRUE or FALSE depending upon whether the process is currently in the simulation queue.

terminated returns either TRUE or FALSE depending upon whether the process is terminated.

Notice

To Our Subscribers

Occasionally, *The C Users Journal* makes its mailing list available to vendors of products we think our readers will find interesting. Current subscribers receive free information in the mail from these vendors.

If you prefer that your name not be used in these mailings, please let us know. Just copy or clip this form and send it with your name and address to:

The **C** Users Journal™

The C Users Journal
1601 W. 23rd. St., Ste. 200
Lawrence, KS 66046-2700

evtime returns the simulation time at which a process is due to be reactivated; *set_evtime* enables a program to change this time.

The *Hold* method removes the active process from the head of the event queue and schedules it to become active a specified number of time units later.

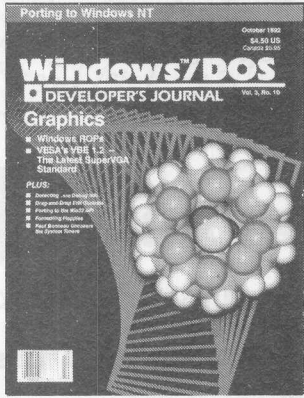
Passivate removes the currently active process from the event queue altogether. If the process is to execute again the program must recreate it.

Cancel removes the process from the simulation queue or simply suspends it indefinitely if it is currently not in the queue.

At any point in time, a process can be in one (and only one) of the following states:

Windows™/DOS

DEVELOPER'S JOURNAL



ADVANCED SERIOUS TECHNICAL

If you're serious about Windows programming, take a look at the only publication for you—the advanced Windows programmer. **Windows/DOS Developer's Journal** is written for experienced programmers who need *detailed, advanced, and up-to-date* technical information about Windows and DOS.

A **FREE** issue of **Windows/DOS Developer's Journal** is yours—call now and ask for a trial subscription. If you like the in-depth programming solutions you find in your **FREE** issue, pay only \$29 for a full year's subscription. If not, write "cancel" on the accompanied invoice and owe nothing. You have *nothing* to lose and a world of knowledge to gain.

FREE TRIAL ISSUE

CALL TODAY!

913-841-1631

FAX 913-841-2624

(Non U.S. orders must prepay \$45 Canada/Mexico
\$64 Outside North America, U.S. funds)

DON'T TYPE IT...

...JUST READ IT

Add a code disk subscription to your magazine subscription and get 12 disks (one each month) of **C Users Journal** listings.

SAVE

- Hours of typing long code
- yourself from entering errors
- 50% off the price of individual disks

The C Users Journal™

Advanced Solutions for C/C++ Programmers

CODE DISK SUBSCRIPTION

Only \$30 (prepaid) a year
\$50 (Non North American)

For more information or to order:

CALL 913-841-1631
FAX 913-841-2624

- active: the process is at the head of the queue maintained by the scheduler (to be described shortly) and its actions are being executed.
- suspended: the process is in the queue maintained by the scheduler, scheduled to become active at a specified time in the future.
- passive: the process is not in the scheduler's queue. Unless another process brings it back into the queue, it will not execute any further.
- terminated: the process is not in the scheduler's queue and has no further actions to execute.

There are five ways to activate a process, and similarly five ways to reactivate a waiting process:

- before another process (*ActivateBefore* and *ReActivateBefore*);
- after another process (*ActivateAfter* and *ReActivateAfter*);
- at a specified (simulated) time (*ActivateAt* and *ReActivateAt*);
- after a specified (simulated) delay (*ActivateDelay* and *ReActivateDelay*);
- activate now (at the current simulated time) (*Activate* and *ReActivate*).

(Note that if a process is already scheduled, reactivation will simply re-schedule the process.)

The *CurrentTime* method returns the current simulation time; programs typically call *CurrentTime* to control action relative to a given time period.

The Simulation Scheduler

As in SIMULA, simulation processes (entities) execute at their assigned simulation time, which is typically determined by an appropriate distribution function. Only one process executes in any instance of real time, but many processes may execute at any instance of simulation time. Programs place currently inactive processes in a simulation queue (the event queue), which is arranged in order of increasing simulation time.

To coordinate the execution of these processes, the scheduler manages the simulation queue as follows: when no process is currently active, the scheduler selects a process to run from the head of the queue and (re-) activates it. When no processes are left to execute, i.e., the queue is empty, the simulation ends.

The simulation queue is organized as a tree to improve the efficiency of the scheduling algorithm. All nodes (processes) at the same level of the tree are assigned to the same simulation time, as shown in Figure 1.

Because the scheduler manages processes in the simulation environment it cannot itself be a simulation process. Like the main thread to be described later, the scheduler is a priority thread within the environment and as such must be controlled in a slightly different manner than the other simulation entities. The structure of the scheduler is extremely simple and is shown as follows:

```
class Scheduler : public LWP_Thread
{
public:
    Scheduler ();
    ~Scheduler ();
```

```

void Body ();
double CurrentTime ();
};

```

Every simulation application must start one scheduler before the simulation can begin. The example to be described near the end of this article illustrates use of the scheduler.

Priority Threads

C++ SIM executes two "priority" threads which cannot be derived from the *Process* base class and therefore must be activated and deactivated separately. These threads are as follows:

- the simulation scheduler: this thread must be activated via the *Resume* method of the thread base class from which it is derived (e.g., *LWP_Thread*);
- the thread associated with *main*. A program must suspend this thread to allow other threads to run since this thread has the highest priority in the system. Calling the *Thread* class *Initialize* method within the main body of the simulation code adds this thread to the thread queue maintained by class *Thread*. The thread's presence in the queue allows the *Suspend* method to act on it when the program requires it to become inactive (using the *Thread::Self()->Suspend()* operation).

Distribution Functions

Simulations often require distribution functions of various events (e.g., the rate of arrivals of jobs at a processor, or the time between failures for a node). C++ SIM provides a set of classes which give access to various useful distribution functions, including the following: *RandomStream*, *UniformStream*, *Draw*, *ExponentialStream*, *ErlangStream*, *HyperExponentialStream*, and *NormalStream*. By creating instances of these classes the simulation processes can gain access to the appropriate distribution function. Figure 3 shows the class hierarchy of the distribution functions.

RandomStream and NormalStream

Classes *RandomStream* and *NormalStream* illustrate how the distribution functions are derived and show how further functions

could be built. *RandomStream* (from which all other distribution functions are derived) is shown as follows:

```

class RandomStream
{
public:
    RandomStream (long MGSeed = 772531L,
                  long LCGSeed = 1878892440L);
    virtual double operator() () = 0;
    double Error ();

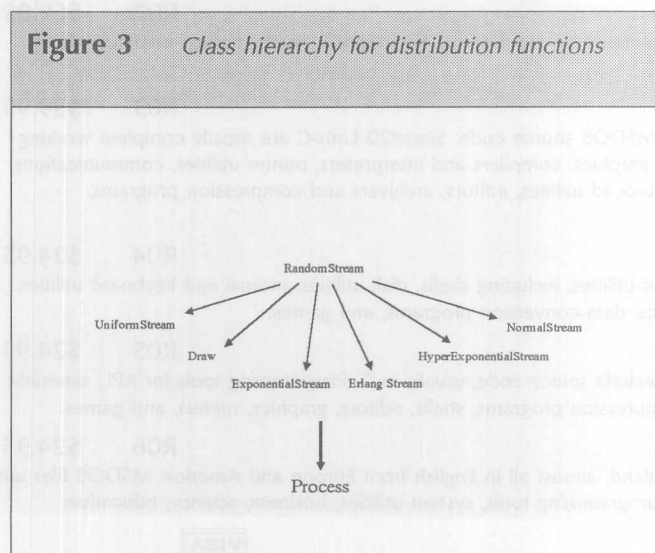
protected:
    double Uniform ();

private:
    double MGen ();
    double series[128];
    long MSeed, LSeed;
};

```

The *Error* method returns a chi-square error measure on the uniform distribution function. The *Uniform* method generates random numbers; *Uniform* uses the linear congruential generator based on the algorithm from [Knuth Vol2], and shuffles the results of the linear generator with the multiplicative generator as suggested by [Knuth Vol2] [3] to obtain a sufficiently uniform random distribution.

Figure 3 Class hierarchy for distribution functions



BDL

Btrieve Development Library

Reduces Development and Maintenance Time

- Replaces Novell's BTRV() interface without loss in processing speed
- Shorten learning curve
- Facilitates writing of consistent code
- Uses the same descriptive names for op codes as the Btrieve documentation
- Behind the scenes allocation and tracking of parameters used to interface with Btrieve
- Macros and Functions are provided to ease file creation
- Function Call to provide Stat information
- Adds functionality and eases use of Extended Calls
- Default Error Handler which uses BIOS to display an error message box
- Customizable Severity Action Table for Error Handling
- Hook to Customize your own Error Handler Display
- Btrieve Files are automatically closed at exit
- Example programs included on disk
- Requires Btrieve for DOS or Brequest for network

- Support for Watcom, Microsoft and Borland C/C++ compilers
- Runs under DOS, Windows and OS/2

To ORDER or
to request a
FREE Demo
Disk Call:

\$200⁰⁰
\$375⁰⁰ with source

30 DAYS
FREE
SUPPORT



INTREPID
Technology
800-398-8383

◆ Request 270 on Reader Service Card ◆


```

class NormalStream : public RandomStream
{
public:
    NormalStream (double Mean, double StandardDeviation);
    virtual double operator() ();

private:
    double Mean, StandardDeviation;
    double z;
};

```

The *operator()* uses the polar method in [Knuth Vol2] [4] to implement the *NormalStream* by making use of the *Uniform* method of *RandomStream*.

SIMSET

C++ SIM also provides entity and set manipulation facilities similar to those provided by the *SIMSET* classes of *SIMULA*. These facilities break down into two classes:

- *Link*: the *Link* class provides elements of a doubly linked list;
- *Head*: the *Head* class maintains doubly linked lists of *Link* elements.

Class *link* is defined as follows:

```

class LINK
{
public:
    virtual ~Link ();

    Link* Suc () const;
    Link* Pred () const;

    Link* Out ();
    void InTo (Head*);

    void Precede (Link*);
    void Precede (Head*);
    void Follow (Link*);
    void Follow (Head*);

protected:
    Link ();
};

```

Because it makes no sense to create instances of *Link* objects, the constructor for *Link* is protected — programs must derive a class from *Link* to benefit from its functionality.



CD-ROM Software

Walnut Creek CDROM



- | | | |
|--|-----|---------|
| • C Users' Group Library | R01 | \$49.95 |
| Volumes 100 to 364. Source code for editors, disassemblers, compilers, interpreters, communications, games, tutorials, math libraries. Most code for MSDOS, much for UNIX and other systems. Disk includes the first three text editions of the CUG Directory: indexing and describing every file and reviewing major packages. | | |
| • Libris Britannica | R02 | \$69.95 |
| Public domain and shareware from PDSL, Sussex, for DOS: extensive sections on electronics, engineering, mathematics, medicine, ham radio, including the entire C Users' Group UK archive. | | |
| • Source Code | R03 | \$39.95 |
| The Usenet archives, Simtel20 Unix-C archives, and a large collection of MSDOS source code. Simtel20 Unix-C are mostly complete working programs for archiving, benchmarks, databases, editors, file management, graphics, compilers and interpreters, printer utilities, communications, networking. MSDOS source is almost 2000 Zipped packages including Autocad utilities, editors, archivers and compression programs, emulators, compilers and interpreters; mostly C source code. | | |
| • CICA MicroSoft Windows | R04 | \$24.95 |
| The entire CICA Windows Collection from Indiana University, hundreds of utilities, including shells, disk utilities, mouse and keyboard utilities, screen savers, backup/restore programs, performance monitors, diagnostics, data conversion programs, and games. | | |
| • SIMTEL20 MSDOS | R05 | \$24.95 |
| The entire Simtel20 MSDOS archive, 640 megabytes in 9000+ files. Many include source code, usually in C. Programming tools for APL, assembly, C, Pascal, Perl, Prolog, Smalltalk, etc. Communications utilities, BBS's, compression programs, shells, editors, graphics, menus, and games. | | |
| • Garbo MSDOS/Mac | R06 | \$24.95 |
| Contains the MSDOS and Mac archives from the University of Vaasa, Finland, almost all in English from Europe and America. MSDOS files are 250 megabytes including programs for animation, archive utilities, BBS's programming tools, system utilities, business, science, education. | | |

To Order Call 913-841-1631 FAX 913-841-2624

✧ Request 251 on Reader Service Card ✧



Suc and *Pred* return the successor and predecessor of this list element respectively. These functions return 0 if no such element exists.

Out removes the object to which it currently belongs (if any) from the linked list. *InTo* makes this object the last element in a linked list if the list exists; If the list doesn't exist *Out* attempts to remove the object from any linked list to which it may belong.

Precede also places an object in a linked list. If *Precede* is passed another *Link* element, say *L*, then if *L* is a member of a linked list, this object is placed into the same linked list and immediately preceding *L*. If *L* isn't a member of a linked list, the result is the same as for *Out*. If *Precede* is passed a *Head* object it produces the same result as *InTo*.

Follow acts similarly to *Precede*, except that *L.Follow(L1)* inserts *L* immediately after *L1*, and *L.Follow(H)*, places *L* as the first element in *H*, where *H* is a *Head* object.

Note that as in SIMULA, *Link* elements can only belong to one linked list at a time.

Class *Head* is defined as follows:

```
class Head
{
public:
    Head ();
    virtual ~Head ();

    Link* First () const;
    Link* Last () const;

    long Cardinal () const;
    boolean Empty () const;

    void Clear ();
};
```

First and *Last* return references to the first and last *Link* objects in the list respectively. If the list is empty then these functions return 0.

Cardinal returns the number of *Link* objects in the list, and *Empty* returns TRUE if the list is empty, FALSE otherwise. *Clear* removes all *Link* objects from the list.

Example: Job Service Simulation

This example is taken from [Mitrani 82] and simulates a process scheduler for a machine which attempts to execute as many process (jobs) as possible. The machine can only process one job at a time and queues job requests until it can deal with them. The machine is prone to failures, so started jobs will be interrupted by such failures and delayed until the machine has been repaired (re-activated), at which point it is forced to restart execution from the beginning (i.e., it is placed at the head of the job queue). The main processes within this example are:

- *Arrivals*: this process controls the rate at which jobs arrive at the service (Machine).
- *Breaks*: this process controls the availability of the Machine by "killing" it and restarting it at intervals drawn from a *Uniform* distribution.

- *Job*: this process represents the jobs that the Machine must process.
- *Machine*: this is the machine on which the service resides. *Machine* obtains *Jobs* from the job queue for the service and then attempts to execute them. The machine can fail and so the response time for *Jobs* is not guaranteed to be the same every time the job is performed.

Arrivals

The Arrivals class definition is relatively simple since none of the other processes invoke operations on it. *Arrivals* is defined as follows:

```
class Arrivals : public Process
{
public:
    Arrivals (double);
    ~Arrivals ();

    void Body ();

private:
    ExponentialStream* InterArrivalTime;
};
```

HYPERCTM Compiler

Unleash the parallelism in your application!

The HyperC compiler provides a **low cost entry** into the field of massively parallel computing by allowing you to:

- 1) **Develop** on your workstation,
- 2) **Distribute** the work among your workstations,
- 3) **Be operational** on parallel computers (3Q 94).

It provides a very cohesive solution to the problem of programming parallel computers and effectively using their available power. While the programming model is **SIMD** (single thread of control, easy debugging) to facilitate the programmer's job, the execution takes place on **MIMD** architectures. The data parallel extensions of the HyperC language to C were designed to:

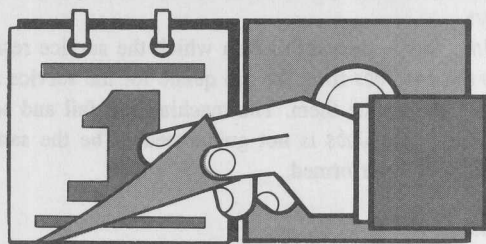
- 1) Increase data locality,
- 2) Minimize communication requirements,
- 3) Overlap communications with computations,
- 4) Permit incremental parallelization of applications.

If you are looking for an easy to learn, yet powerful, language that focuses on the parallelism of applications and not of target architectures, contact our U.S. distributor:

Fortunel Systems, Inc. - 1135 Kildaire Farm Rd. - Suite 311-5
Cary, NC 27511 - Tel: 919-319-1624 - Fax: 919-319-1749.



◆ Request 312 on Reader Service Card ◆



Calendar of Events

April

18-21 ISCA '94: 21st International Symposium on Computer Architecture. Chicago, IL. Sponsored by ACM special interest group ARCH and the IEEE-TC CA. Contact Wen-mei Hwu, Coordinated Science Laboratory, 1308 W. Main Street, Urbana, IL 61801. (217)-244-8270. e-mail: hwu@crhc.uiuc.edu.

19-21 Embedded Systems Conference East. Boston, MA. Contact Amy Royer, Embedded Systems Conference, 600 Harrison St., San Francisco, CA 94107. (415)-905-8198.

24-28 CHI '94: ACM Conference on Human Factors in Computer Systems. Boston, MA. Sponsored by ACM special interest group CHI. Contact Thomas Hewett, Drexel University, Dept. of Psych/Soc/Anthro, Central Receiving, 33rd & Ludlow Streets, Philadelphia, PA 19104. (215)-590-8616, 8672. e-mail: hewett.chi@xerox.com.

May

17-20 CoopIS-94: Second International Conference on Cooperative Information Systems. Formerly "Intelligent & Cooperative Information Systems (ICICIS)." Toronto, Canada. Contact John Mylopoulos, (416)-978-5180. e-mail: coopis@cs.toronto.edu.

23-25 Symposium on Theory of Computing. Montreal, Canada. Sponsored by ACM special interest group ACT. Contact Pierre McKenzie, Dept. IRO, Universite de Montreal, C.P. 6128 succursale A Montreal, Quebec H3C 3J7 CANADA. (513)-343-6176. e-mail: mckenzie@iro.umontreal.ca.

June

9-11 The Grace Hopper Celebration of Women in Computing. Washington, D.C. Sponsored by ACM and CRA, the Computing Research Association. Contact Anita Borg, Digital Equipment Corp., 250 University Ave., Palo Alto, CA 94301. (415)-688-1367. e-mail: borg@pa.dec.com.

25-26 5th International Workshop on ML and its Applications. Orlando, FL. Sponsored by ACM special interest group PLAN. Contact John Reppy, AT&T Bell Laboratories, 600 Mountain Avenue, Rm. 2A-428, Murray Hill, NJ 07974. (908)-582-4084. e-mail: jhr@research.att.com.

26-29 6th Annual ACM Symposium on Parallel Algorithms and Architectures. Cape May, NJ. Sponsored by ACM special interest groups ACT and ARCH. Contact Satish Rao, NEC Research Institute, 4 Independence Way, Princeton, NJ 08540. (609)-951-2714. e-mail: satish@research.nj.nec.com.

The constructor initializes the stream from which the rate of *Job* arrivals is drawn and the destructor simply cleans up before the object is destroyed:

```
Arrivals::Arrivals (double mean)
{
    InterArrivalTime = new ExponentialStream(mean);
}
Arrivals::~Arrivals () { delete InterArrivalTime; }
```

The main body of *Arrivals* (shown below) simply waits for an amount of time dictated by the rate of arrivals stream and then creates another *Job*. This procedure is repeated until the simulation ends.

```
void Arrivals::Body ()
{
    for (;;) // infinite loop
    {
        Hold((*InterArrivalTime)());
        Job* work = new Job();
    }
}
```

Job

Unlike *Arrivals*, which is an active entity within the simulation, the *Job* class does not need to be a separate process, since it is simply enqueued when it is created and dequeued by the *Machine* when it can be executed. All a given *Job* must do is calculate how long it took to be "processed":

```
class Job
{
public:
    Job ();
    ~Job ();

private:
    double ArrivalTime;
    double ResponseTime;
};
```

Because no operations are invoked on instances of the *Job* class, its constructor and destructor perform all of its work:

```
Job::Job ()
{
    boolean empty;

    ResponseTime = 0;
    ArrivalTime = sc->CurrentTime();
    empty = JobQ.IsEmpty();
    JobQ.Enqueue(this); // place this Job on to the queue
    TotalJobs++;

    if (empty && !M->Processing() && M->IsOperational())
        M->Activate(); // Machine idle as no Jobs in queue
```



```

// and not broken
}

Job::~Job ()
{
    ResponseTime = sc->CurrentTime() - ArrivalTime;
    TotalResponseTime += ResponseTime;
}

```

Queue

The program places jobs which are not being serviced in a job queue. As with the *Job* class, an instance of *Queue* is not required to be active, and as such *Queue* is not derived from the *Process* class.

Queue is defined as follows:

```

class Queue
{
public:
    Queue ();
    ~Queue ();

    boolean IsEmpty ();
    // returns TRUE if no Jobs in queue
    long QueueSize ();
    // returns number of Jobs in queue
    Job* DeQueue ();
    // returns head of queue
    void Enqueue (Job*);
    // places Job at tail of queue
};

```

Machine

The *Machine* process obtains *Jobs* from the queue and processes them. Since *Machine* is prone to failures *Jobs* can take extended periods of time to complete. Other simulation processes invoke various operations on the machine (for example to determine whether or not it has failed):

```

class Machine : public Process
{
public:
    Machine (double);
    ~Machine ();

    void Body ();

    void Broken ();
    void Fixed ();
    boolean IsOperational ();
    boolean Processing ();
    double ServiceTime ();

private:
    ExponentialStream* STime;
    boolean operational;

```

```

    boolean working;
};

```

As with the *Breaks* and *Arrivals* processes, *Machine's* constructor and destructor initialize and delete the stream that dictates the time required to process a *Job*.

Processing returns the current status of the machine, i.e., whether or not it is executing a job:

```
boolean Machine::Processing () { return working; }
```

Broken and *Fixed* de-activate (crash) and re-activate the machine respectively:

```

void Machine::Broken () { operational = false; }
void Machine::Fixed () { operational = true; }

```

IsOperational indicates whether or not the machine is currently active (i.e., whether it has "crashed"):

```
boolean Machine::IsOperational () { return operational; }
```

ServiceTime returns the time required to service a given job based on the relevant distribution function initialized within the constructor:

FREE
Evaluation
Package Available

The Heart of the Matter...

RTXC™

Real-Time Multitasking Executive

o INTEL 80x86/x86, 80x96, 80x51
o HITACHI 6303, H8/5xx

o MOTOROLA 680x0, 683xx, EC0x0, 68HC11, 68HC16
o NEC V20/V25/V53
o SIEMENS 165/166/167
o TI C3x
o ZILOG Z80/Z180

- Preemptive Scheduling
- Fixed or Dynamic Priorities
- Timeout on some services
- Configurable and ROMable
- Intertask Communications
 - Messages
 - Queues
 - Semaphores
- Memory Management
- Resource Manager
- Over 65 Executive Services Available
- System Level Debugging Utility Included
- File Manager Now Available

- System Generation Utility
- Written in C
- Source Code Included
- No Royalties
- Technical Support
- Broad C Compiler Support
- Sensible License Agreement
- 600 Page User's Manual
- 3 Configurations Available

One Time License Fee From \$995

Discounts for Multiple Licenses/Ports™

The only real-time kernel you'll ever need™

Embedded System Products, Inc.

11501 Chimney Rock, Houston, TX 77035

FAX 713-728-1049

Phone 800-525-4302 or 713-728-9688

TCPIP
AVAILABLE

◆ Request 380 on Reader Service Card ◆

```
double Machine::ServiceTime () { return (*STime)(); }
```

The main body of the *Machine* gets a *Job* from the job queue (if one is available) and attempts to process it before looping again:

```
void Machine::Body ()
{
    for (;;)
    {
        working = true;

        while (!JobQ.IsEmpty())
            // continue as long as Jobs are available
            {
                Hold(ServiceTime());
                Job* J = JobQ.Dequeue();

                ProcessedJobs++;
                // keep track of number of completed Jobs
                delete J;          // remove finished Job
            }

        working = false;
        // no Jobs in queue so become idle
        Cancel();
    }
}
```

```
}
}
```

Breaks

The *Breaks* class defines a process which simply waits for a specific period of time before "killing" the *Machine* process. This process then waits again before re-activating the machine. The *Breaks* class definition is relatively simple:

```
class Breaks : public Process
{
public:
    Breaks ();
    ~Breaks ();

    void Body ();

private:
    UniformStream* RepairTime;
    UniformStream* OperativeTime;
    boolean interrupted_service;
};
```

The constructor and destructor simply initialize and delete the streams used by the *Breaks* process respectively.

The main body of the *Breaks* process activates and deactivates the *Machine* process. The *Machine* fails and recovers according to the *OperativeTime* and *RepairTime* distribution functions respectively. The body is defined as follows:

```
extern Machine* M; // This is the machine used to
                  // service requests
extern Queue JobQ; // This is the queue from which
                  // Jobs are drawn

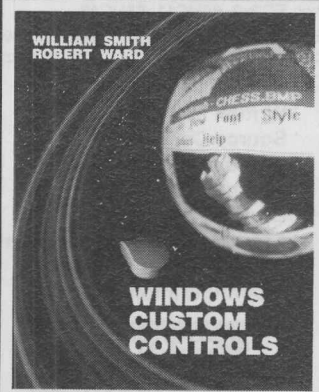
void Breaks::Body ()
{
    for (;;)
    {
        Hold((*OperativeTime)());
        M->Broken();
        // de-activate the Machine
        M->Cancel();
        // remove Machine from Scheduler queue

        if (!JobQ.IsEmpty())
            interrupted_service = true;

        Hold((*RepairTime)());
        M->Fixed(); // re-activate the Machine
        if (interrupted_service)
        {
            interrupted_service = false;
            M->ActivateAt(M->ServiceTime() +
                          CurrentTime());
        }
        else
    }
}
```

CUSTOMIZE YOUR WINDOWS! For Experienced Programmers— GIVE WINDOWS YOUR OWN DISTINCTIVE LOOK.

WILLIAM SMITH
ROBERT WARD



**WINDOWS
CUSTOM
CONTROLS**

Extend the Windows API
with your own components

Build replacements for
standard controls

Create controls that handle
huge amounts of text

Learn how to

- create a toolbox class
- create a custom dialog class
- interface custom controls to
the Dialog editor

**WINDOWS
CUSTOM
CONTROLS**

William Smith
Robert Ward

This book includes all the
code for a powerful set of
ready-to-use custom controls.
You can modify the code for
your own design.

Compiles under Microsoft
C/C++, Quick C, and Borland
C++ and has been tested
under Windows 3.1

Companion Disk for \$29.95

ORDER TODAY!

Order: Book W99, Disk W99d,
Book and Disk W99c

913-841-1631

FAX 913-841-2624



**\$ Only
39.95**
\$69.90 with disk
plus shipping



◆ Request 278 on Reader Service Card ◆

```

        M->ActivateAt();
    }
}

```

MachineShop

The *MachineShop* class is the core of the simulation; it starts up all of the main processes involved, and when the simulation ends it prints out the results.

```

class MachineShop : public Process
{
public:
    MachineShop ();
    ~MachineShop ();

    void Body ();
    void Await ();
};

```

The *Body* method starts up the other processes, such as the *Machine*, and then waits until the number of processed *Jobs* is at least 100,000:

```

void MachineShop::Body ()
{
    sc = new Scheduler();           // create the simulation
                                   // queue scheduler
    Arrivals* A = new Arrivals(10);
    M = new Machine(8);
    Job* J = new Job;
    Breaks* B = new Breaks;

    // activate the relevant simulation processes

    B->Activate();
    A->Activate();
    sc->Resume();                   // start up the scheduler
                                   // - it is not a process

    while (ProcessedJobs < 100000)
        Hold(10000);

    cout << "Total number of jobs processed "
    << TotalJobs << endl;
    cout << "Total response time " << TotalResponseTime << endl;
    cout << "Avge response " <<
    (TotalResponseTime/ProcessedJobs) << endl;
    cout << "Avge number jobs present "
    << (JobsInQueue/CheckFreq) << endl;

    // end simulation by suspending processes

    sc->Suspend();
    A->Suspend();
    B->Suspend();
}

```

It isn't necessary to explicitly activate the *Machine* process because the *Breaks* or *Jobs* process will do this.

The *Await* method suspends the thread associated with *main*, thus allowing the other simulation threads to execute:

```

void MachineShop::Await()
{
    Resume();
    Thread::Self()->Suspend();
}

```

Main

The main part of the simulation code initializes the various thread-specific variables used (e.g., the maximum priority of a thread), creates the main body of the simulation code (in this case *MachineShop*) and then suspends the thread associated with *main*:

```

void main ()
{
    LWP_Thread::Initialize();

    MachineShop m;
    m.Await();           // Suspend main's thread
                        // (NOTE: this MUST be done
                        // by all applications).
}

```

Programming Seminars

- ◆ C++ for C Programmers
- ◆ OOA/OOD
- ◆ Advanced C++
- ◆ Other Courses Available...



- Public and On-Site Seminars



- Call for Brochure
- 612.452.3487

Intertech, Incorporated
Borland Assurance
 Authorized Training Center

◆ Request 318 on Reader Service Card ◆

}

Conclusions

The authors of C++ SIM have endeavoured to provide a simulation package which provides similar functionality to that of SIMULA, since SIMULA has fulfilled the needs of users over many years. From their experiences of using SIMULA, both as a general programming language and as a simulation tool, they believe they have been successful. As a result of using C++ they also believe that they have produced a simulation package having several advantages over SIMULA, for example:

- performance — C++ compilers typically generate code that is several times more efficient than similar SIMULA code, and as a result, simulations execute correspondingly faster;
- C++ provides more extensive object-oriented features than SIMULA, allowing, for example, class instance variables to be either publicly or only privately available. In SIMULA, everything is public, affecting the way code is written and providing extra problems during debugging.
- C++ SIM incorporates inheritance throughout its design to an even a greater extent than is already provided in SIMULA. For example, C++ SIM's I/O facilities, random number generators, and probability distribution functions are entirely object-oriented, relying on inheritance to specialize their behavior. Hence, users can add new functionality (e.g., new random

number generators) with little effect on the overall system structure.

Acknowledgements

The authors would like to thank Professor Isi Mitrani for the help he has given them in the development of this simulation package and the time he has devoted to listening to their thoughts and problems. They would also like to thank Ron Kerr for his help with the SIMULA language, and Dr. Graham Parrington for his comments on drafts of this article. The work reported here has been supported by SERC/MOD Grant GR/H81078 and Esprit Broadcast (Basic Research Project Number 6360). □

References

- [Birtwhistle 73] G. M. Birtwhistle, O-J. Dahl, B. Myhrhaug, K. Nygaard, *Simula Begin*, Academic Press, 1973.
- [Black 86] A. Black et al, "Object Structure in the Emerald System", *Proceedings of the ACM Conference on Object-Oriented Programming Systems, Languages, and Applications*, October 1986.
- [Dahl 70] O-J. Dahl, B. Myhrhaug, K. Nygaard, "SIMULA Common Base Language," Technical Report S-22, Norwegian Computing Centre, 1970.
- [Knuth Vol2] Knuth Vol2, *Seminumerical Algorithms*, Addison-Wesley: p. 117.
- [McCue 92] D. L. McCue and M. C. Little, "Computing Replica Placement in Distributed Systems," *Proceedings of the 2nd Workshop on the Management of Replicated Data*, November 1992: pp. 58-61.
- [Mitrani 82] I. Mitrani, *Simulation Techniques for Discrete Event Systems*, Cambridge University Press, Cambridge, 1982: p. 22.
- [Sedgewick 83] R. Sedgewick, *Algorithms*, Addison-Wesley, Reading MA, 1983: pp. 36-38.
- [Stroustrup 86] B. Stroustrup, *The C++ Programming Language*, Addison Wesley: 1986.

Footnotes

- [1] The software to be described in this paper is available via anonymous ftp from arjuna.nc1.ac.uk
- [2] The authors would like to thank Professor I. Mitrani for his help in developing the multiplicative generator used in the simulation. It is based on the following algorithm: $Y[i+1] = Y[i] * 5^5 \bmod 2^{26}$, where the period is 2^{24} , and the initial seed must be odd.
- [3] As suggested by Maclaren and Marsaglia.
- [4] Due to Box, Mullers and Marsaglia



LESS

MEMORY • Under 35K non-discardable memory per instance.
TIME • Loads in a fraction of the time needed by other editors.
MOVEMENT • Nearly all sophisticated features available from either the keyboard or the mouse.
MEMORIZATION • *SpexEdit* uses the same robust set of editing primitives as the Unix Vi editor which are combined to form editing phrases. Other editors have a cumbersome set of prepackaged functions. *SpexEdit* delivers an editing language to be used on the fly.
MONEY • *SpexEdit* is the most-affordable full-feature professional Windows editor for programmers.

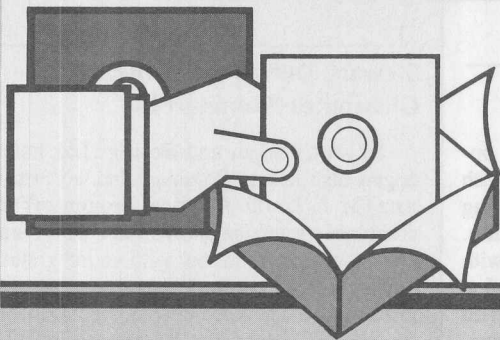
Is MORE!

SpexEdit PROFESSIONAL \$99
 WINDOWS EDITOR
CALL FOR FREE DEMO DISK!
REQUIRES WINDOWS 3.1


Little Wing

4618 J.F.K. BOULEVARD, SUITE 176
 NORTH LITTLE ROCK, ARKANSAS 72116
 CALL (501) 771-2408 9AM-5PM (Central)

◆ Request 119 on Reader Service Card ◆



New Products

Industry-Related News & Announcements

Blue Sky Releases Code Generator for OWL v2.0

Blue Sky Software Corporation has released WindowsMAKER Professional v5, a Prototyper and C/C++ code generator for Windows and Windows NT, which lets Borland C++ v4.0 users generate OWL v2.0 code. WindowsMAKER v5 integrates into the Borland IDE and offers 50 features in addition to those offered by AppExpert.

Features of WindowsMAKER v5 include: Visual Basic-like code editing, toolbar, and context-sensitive SmartMenus; a Visual Screen Designer, which includes drag-and-drop editing and a configurable tool palette; and a Preview Mode which lets users preview and test run an application before compiling. WindowsMAKER v5 also includes Switch-It Code Generation Modules (SIMs), which let users switch to any language or major C++ library during the development of an application. Users can choose to automatically generate ANSI C, MFC and OWL from one design. Using WindowsMAKER v5, users can start a project and decide the type of code to use later and target Windows, Win32s, and Windows NT.

Other features of WindowsMAKER v5 include: toolbar support; application templates; code generation for online help; special effect support for fonts, colors, 3D buttons, patterns, and 256-color bitmaps; style and property setting support for controls; MDI application support; and Extended Functionality Modules (EFMs).

WindowsMAKER Professional v5 includes the ANSI C SIM and is \$995. Registered users of WindowsMAKER Professional v4 can upgrade for \$269. For more information contact *Blue Sky Software Corporation*, 7486 La Jolla Blvd., Suite 3, La Jolla, CA 92037, (800) 677-4946 or (619) 459-6365; FAX: (619) 459-6366.

National Information Systems Releases ACCENT STP

National Information Systems, Inc. has released ACCENT STP (Sun Transition Pack) v2.0, a source code translator for Sun

Select Software Upgrades C++ Designer

Select Software Tools has upgraded C++ Designer, an MS-Windows-based object oriented analysis and design tool, which includes Microsoft's MFC2 Library and the Borland Owl Library. Based on Rumbaugh's Object Modeling Technique (OMT), C++ Designer lets the user structure systems graphically, adding classes and relationships using the GUI. The most recent release allows users to select from classes available in the Microsoft library or from their own class library. Header files for C++ can be generated from the design and linked into the Visual C++ environment.

Features of C++ Designer include: Windows CUA compliance; MDI interface; a data dictionary browsing capability; access control via user-ID; export facility to clipboard or to Windows metafile; project administration including versioning, backup, restore, and off-line; and multi-page printing. C++ Designer also supports the object-oriented and C++ features including: single and multiple-class inheritance; class properties such as attributes and operations; asso-

ciations including multiplicity, roles, qualifiers, and specialized forms such as aggregations; defining the access, virtual and static nature of attributes and operations, and the access and virtual nature of base classes; and storage of additional descriptive and constraint information for classes, attributes, operations, and associations.

C++ Designer can be integrated with Borland's C++ and Turbo C++, and Microsoft's Visual C++. It can also be configured to integrate with other Windows IDEs. Code frames can also be generated from C++ Designer. Code is compatible with most C++ compilers including Borland, Microsoft, and Clarion TopSpeed.

C++ Designer is \$295 per user, including free technical support via a toll free number. Upgrades are sold separately. Users who have purchased C++ Designer within the past three months will receive the MFC2 library at no cost. For more information contact *Select Software Tools, Ltd.*, 1526 Brookhollow Dr., #84, Santa Ana, CA 92705, (714) 957-6633; FAX: (714) 957-6219.

developers who plan to migrate their OPEN LOOK applications to Motif and the Common Desktop Environment. ACCENT STP supports Motif v1.2 and X11R5 on Sun OS v4.1.x, and Solaris v2.x. According to NIS, ACCENT STP will translate 80% to 100% of the C/C++ application source code produced by XView, OLIT, or Devguide GIL files, including header files, where there are equivalent resources offered in Motif. The translated output will be recognizable Motif C or C++ source code.

Features of ACCENT STP v2.0 include support for "drag and drop," TTY widgets, and internationalization support. Other features of ACCENT STP include: compliance to the Motif standard, compatibility with Solaris v2.4/COSE CDE, and application portability to multiple hardware platforms.

ACCENT STP consists of four optional modules. The Devguide Conversion, XView

Conversion, and OLIT Conversion modules are \$4,995 each. The WindowMaker GUI Editor is \$1,495. Motif consulting and training are also available. ACCENT ToolKit, a Motif library and OPEN LOOK ToolKit which helps support OPEN LOOK users after the source code is translated, is available for \$2,495 when purchased with ACCENT STP. For more information contact *National Information Systems, Inc.*, 4040 Moorpark Ave., Suite 200, San Jose, CA 95117, (800) 441-5758; FAX: (408) 246-3127; e-mail: info@nis.com.

Inmark Releases zApp Interface Pack

Inmark Development Corporation has released the zApp Interface Pack, an add-on product to its C++ class libraries. zApp is a portable



StratosWare Releases MemCheck for ANSI and K&R Platforms

StratosWare Corporation has released an ANSI-compliant source code version of MemCheck, its error-detection tool for C/C++. MemCheck can be used for development projects built with ANSI or K&R compliant C/C++ compilers.

MemCheck for ANSI and K&R can detect memory overwrites and underwrites, memory leaks, heap corruption, and out-of-memory conditions, and requires no source code changes for most projects. MemCheck integrates with existing C/C++ code and works at run time to identify errors by source file and line number. The error messages may be directed to the screen, written to log files, or sent in network or e-mail messages.

MemCheck for ANSI and K&R includes source code and requires no debugging in-

formation and no special compilation options. Developers may ship applications with MemCheck linked in, royalty-free, allowing detection of errors at beta or customer sites. According to StratosWare, applications with MemCheck linked in run unchanged and at full speed. MemCheck may be switched on or off at run time, linked out via a "Production" library, or compiled completely out with no source code changes.

MemCheck for ANSI and K&R is \$199. StratosWare offers free technical support via fax, CompuServe, Internet, and a toll free number. For more information contact *StratosWare Corporation, 1756 Plymouth Rd., Suite 1500, Ann Arbor, MI 48105, (313) 996-2944; FAX: (313) 747-8519.*

C++ Application Framework that gives application developers cross-platform portability through object-oriented C++ classes. zApp v2.0 is shipping on Windows, Windows NT, DOS Test, DOS Graphics, and OS/2.

The zApp Interface Pack augments zApp by providing a set of additional classes that add graphical elements to applications. Classes in the zApp Interface Pack (ZIP) provide applications with toolbars and status lines, 3D custom controls, bitmap buttons, and a table object.

Quoting Howard Love, President and CEO of Inmark, "Developers can now incorporate the most advanced features of modern applications with just a few lines of code. Toolbars, spreadsheet-like tables, and other high-level controls, which would normally require weeks or months of engineering time, are now available to developers with just a few lines of code."

Demonstration software of zApp is available on the Inmark BBS. For more information contact *Inmark Development Corporation, 2065 Landings Dr., Mountain View, CA 94043, (415) 691-9000; FAX: (415) 691-9099; BBS: (415) 691-9990, CompuServe GO INMARK*



Quadralay Ship UDT for C/C++

Quadralay Corporation has begun shipping UDT for C/C++ v1.2, an open UNIX development environment. UDT for C/C++ works with existing tools and code base, and acts as an incremental front end to existing compilers and other development tools. Features of UDT for C/C++ include: source code browsing, editing, project manage-

ment, compiling, and prototyping for C++ classes and libraries.

UDT for C/C++ v 1.2 supports SPARC SunOS v4.1.x, SPARC Solaris v2.x, Intel SCO Open Desktop, Intel Solaris v2.x, HP 9000/700, and the RS/6000. UDT for C/C++ v1.2 single and multiple licenses are \$595 and \$795 per seat respectively. A free, thirty-day evaluation with online documentation is available via anonymous ftp from *ftp.quadralay.com* (192.195.32.1) or by request. For more information contact *Quadralay Corporation, 8920 Business Park Dr., Austin TX 78759, (512) 346-9199; FAX: (512) 794-9997; e-mail: info@quadralay.com.*



Sector Seven Releases MakeMaster v2.6

Sector Seven has released MakeMaster v2.6. MakeMaster (formerly DEPGEN), reads C source code and include files to create makefiles. Features of MakeMaster v2.6 include: support for C++, an optimized file search algorithm, flat dependency lists, relative directory references, and custom compiler command lines.

MakeMaster v2.6 supports multiple disk and directory projects, cross-compilers and linkers, libraries, and DLLs. MakeMaster v2.6 runs under DOS v3.3+, and supports ANSI C and C++, Borland's Turbo Make, and Microsoft's NMake. MakeMaster v2.6 is \$49.95. For more information contact *Sector Seven, P.O. Box 11391, Burke, VA 22009, (703) 866-9477.*



Stewart, Dufour and Gossage Distributes ProminareTM

Stewart, Dufour and Gossage Ltd. have begun distributing Prominare Inc.'s ProminareTM in North America. ProminareTM combines a graphically oriented GUI design and code generation tool with an integrated development environment and a programming editor. The GUI development part of ProminareTM supports PM controls including those for MMPM/2 and Pen-PM. ProminareTM supports C/C++ compilers and CommonView. The GUI tool also supports direct creation of resource files as well as programming code.

According to the company, the integrated development environment of ProminareTM eliminates most of the common mistakes encountered in trying to build OS/2 applications. ProminareTM's graphical interface supports compiler-independent specification of common options required for compiling and linking programs. The IDE works with the programming editor to manage compiler and linker-detected errors.

ProminareTM's programming editor is PM-based but performs, according to the company, like a character-based editor. The editor is integrated with the IDE and the GUI design tool and provides links to on-line help for the IBM Toolkits. The single-user and network versions of ProminareTM are \$895 each. Additional workstation modules are \$795. There are no run-time fees or royalties. For more information contact *Stewart, Dufour & Gossage Ltd., 210-1730 Courtwood Cr., Ottawa, Ontario, K2C 2B5, Canada, (613) 225-2121; FAX: (613) 225-2624; BBS: (613) 225-2968.*



DDC International A/S Introduces 1st OBJECT EXEC

DDC International A/S has introduced 1stOBJECT EXEC, a C++-based real-time operating system aimed at industrial use. DDC-I regards C++ as a highly suitable candidate for safely solving control problems, and expects its run-time system to find uses from electronic measuring systems to life and safety-critical applications. According to DDC-I, their C++ run-time system can facilitate rapid development of complex systems and advanced communications with the switches, sensors, and motors used in new designs.

For more information contact *DDC International A/S, Gi. Lundtoftevej 1B, DK-2800 Lyngby, Denmark, +45 45 87 11 44; FAX: +45 45 87 22 17; Telex: 37704 ddc dk.*

Omega Systems Ships VERSIONS

Omega Systems has begun shipping VERSIONS, a version control system for Windows. VERSIONS provides version control support for programmers by managing multiple project resources including source code, documentation, or other files. VERSION supports varied file formats, including binary files, and doesn't limit the number or types of files which can be maintained. VERSIONS is network-compatible and supports multiple developers working on the same project, allowing storage of both temporary and permanent versions of a file.

Eschewing the approach of storing incremental changes to files, VERSIONS stores previous versions of a file in a "master project," a proprietary format that labels, maintains, and tracks files for access to both the most current, as well as any past version. VERSIONS also tracks changes to a project either on a server or local workstation through the master project. Users check files into and out of the master project as needed and VERSIONS automates the process of determining which files need to be checked in or out. VERSIONS is \$99. One copy of VERSIONS is required for each workstation in a networked installation. For more information contact *Omega Systems, 5405 Alton Parkway, Suite 5A494, Irvine, CA 92714, (800) 458-5467 or (714) 253-6700; FAX: (714) 253-6712.*

EMS Professional Shareware Upgrades C/C++ Utility Library

EMS Professional Shareware has upgraded its C/C++ Utility Library on CD-ROM. The library has over 1000 public domain and shareware products for programmers using C/C++, Microsoft C, and Turbo C. The products are compressed on 46 1.44 Mb diskettes or one CD-ROM. All products in the library, and 150 commercial products, are described in an indexed database which accompanies the library. When a programmer needs to locate a particular type of C/C++ product, he or she can find it by vendor, name, type, release date, or free text search across descriptions. The C/C++ Utility Library contains a variety of files, including: Array, Binary Tree, Communication, Compression, Database, Debugger, Editor, Graphics, Linked List, Memory Management, MS Windows, MS Windows NT, Paradox Engine, Program Generator, Reference, Spreadsheet, String, TSR/ISR, Virus, and other types.

The C/C++ Utility library is \$59.50 on CD-ROM or \$149 for the diskette versions.

Network Dynamics Upgrades Internationalization Toolkit

Network Dynamics has released the Internationalization Toolkit v3.0. The toolkit is used by programmers to simplify and expedite the internationalization/localization of software. Formerly known as "The String Externalization Tools," the toolkit has been expanded to include support for Windows and Presentation Manager string resource files, string length tracking, string file appending, an extraction "undo" utility, string replacement, handling of static initialization, and a graphical user interface.

Version 2.0 features like string extraction, multi-byte Kanji support, software life-cycle support, string tagging, string caching, and string re-insertion have been expanded in version 3.0, with the intent of providing

greater flexibility and ease of use. The user's manual has also been expanded. A "white paper" on the subject of software internationalization is available as an option. Sample programs are included to illustrate DOS code page swapping and keyboard manipulation.

The Internationalization Toolkit is available in source code form for C/C++. Plans call for release of Turbo Pascal and Quick Basic versions in 1994. The Internationalization Toolkit supports DOS, OS/2, and UNIX platforms. The Internationalization Toolkit is \$249.95 for a royalty-free source code license for up to 10 programmers. For more information contact *Network Dynamics, (804) 220-8771; FAX: (804) 220-5741.*

A subset of the C/C++ library containing 406 files for C++ only is available on 20 diskettes for \$59.50. All products come with a 30-day, money-back guarantee. For more information contact *EMS Professional Shareware, 4505 Buckhurst Ct., Olney, MD 20832, (301) 924-3594; FAX: (301) 963-2708; e-mail: eengelmann@worldbank.org.*

Cadre Technologies Introduces Ensemble Viewer

Cadre Technologies Inc. has released Ensemble Viewer, an interactive 2-D and 3-D graphical tool for visualizing C programs.

Ensemble Viewer supports browsing of program flow, data structure, and physical file structures. Ensemble Viewer provides interactive views of key program aspects by displaying program information and test results that are stored in the Ensemble database. Ensemble Viewer lets the user interact with the software design, code, and files by viewing a graphical representation of the actual program structures. This interaction lets a user understand the program or see the impact of program changes without having to read through the source code. Ensemble Viewer is available for Sun SPARCstations and the company plans called for a release on HP9000 and IBM RS/6000 during the first quarter of 1994. Prices for Ensemble Viewer start at \$2,400 depending on configuration. For more information contact *Cadre Technologies Inc., 222 Richmond St., Providence, RI 02903, (401) 351-5950; FAX: (401) 351-7380.*

TerraLogics Upgrades Mapping Software Toolkit

TerraLogics, Inc. has upgraded TerraView, its geographic mapping software development kit. TerraView v4.0, lets developers embed geographic maps directly into the source code of Windows applications developed with Microsoft's Visual C++ and C/C++ v7, Borland's C++ v3.1, Gupta's SQL Windows, and Powersoft's PowerBuilder.

Features of TerraView v4.0 include: raster data support, which lets developers superimpose TerraView data maps over aerial photographs or satellite images; style sheets, which are used to customize map displays; dynamic renditioning, which supports style changes for a single use of a map; and faster updating of mobile symbols for real-time tracking and display applications. TerraView v4.0 lets map users access data from many database managers, including Gupta SQLBase, dBASE, and Oracles, as well as Defense Mapping Agency's Digital Chart of the World and the Census Bureau's Topologically Integrated Geographic Encoding and Referencing (TIGER) data.

TerraView v4.0 is available for MS-DOS and Windows, Apple Macintosh, and UNIX systems from SUN, HP, and IBM. For more information contact *TerraLogics, Inc., 600 Suffolk St., Lowell, MA 01854, (508) 656-9900; FAX: (508) 656-9999.*

PractiSys Releases STORC GOLD v2.0

PractiSys has released v2.0 of STORC GOLD, its Windows form conversion tool.



Computer Applications Specialists Introduces ProMet

Computer Applications Specialists has introduced ProMet, a C/C++ metrics utility. ProMet measures program size, number of comments and comment density, McCabe's and other complexity measurers, level of nesting, number of program jumps, and number of literals. These measures provide quantitative data for estimation of effort and conformance to programming style guidelines.

In addition to calculation of metrics for each function and each module, ProMet provides program summary data and a "Top Ten" report that highlights the ten functions that have the highest metric scores. The Top

Ten functions are presented both in a file list and as a graph to highlight the code sections that have the highest likelihood of having errors. The Top Ten list lets testing and review resources focus on code areas that are the most complex and may suggest the need for additional oversight.

ProMet is \$99 and comes with a manual that provides a background on software metrics and their use. For more information contact *Computer Applications Specialists*, 9948 Hibert St., Suite 103, San Diego, CA 92131, (619) 695-2600; FAX: (619) 695-0794.

STORC GOLD v2.0 lets Microsoft Visual C++ developers directly import Visual Basic .FRM files into C++ projects, reusing both form design and VBX controls.

STORC GOLD v2.0 is \$45.93 per copy per single user. Developers can download a fully-functional evaluation copy from CompuServe (GO WINSK or GO MSBASIC), or can obtain a copy from PractiSys for \$3. For more information contact *PractiSys*, 4767 Via Bensa, Agoura, CA 91301, (818) 706-8877.



Digital Information Systems Corporation Ports PVCS to Alpha AXP

Digital Information Systems Corporation and Digital Equipment Corporation have ported Intersolv's PVCS Version Manager v5 and PVCS Configuration Builder v5 products to Digital's Alpha AXP. Under the terms of the agreement, DISC is porting, distributing, and providing technical support for PVCS products on operating systems not supported by Intersolv. DISC has completed ports to OpenVMS, VAX/VMS, OSF/1, and several UNIX-based platforms.

Features of PVCS Version Manager v5 include: SQL support facilities, expanded file support, user-defined delta generation, and internationalization capabilities. Features of PVCS Configuration Builder v5 include: run-time diagnostics, build techniques, directives, and build script compilation capabilities.

The price for PVCS Version Manager v5 starts at \$599 for a 1-4 user license and Configuration Builder starts at \$399. For more information contact *Digital Information Systems Corporation*, 11070 White Rock Rd., Rancho Cordova, CA 95670, (800) 366-3472 or (916) 635-7300; FAX: (916) 635-6549.



Odyssey Development Releases ISYS Search Engine

Odyssey Development, Inc., has released the ISYS Developers' Toolkit, which includes Odyssey's ISYS text retrieval software. Developers and OEMs can integrate the ISYS text retrieval engine into applications for CD-ROM authoring, electronic publishing, or document and image management. ISYS can access textual information stored in word processor and other files. ISYS can read 28 word processor formats, as well as some spreadsheet and database files. Documents to be searched remain in their native formats. The Developers' Toolkit also lets OEMs develop External Access Modules. ISYS can then access and index "foreign" data sources, such as text stored in a relational database.

The ISYS engine is available for Microsoft Windows and MS-DOS. The ISYS engine can be called from many major languages. Sample code is provided for C, Pascal, and Visual Basic. For more information contact *Odyssey Development, Inc.*, 650 S. Cherry St., #220, Denver, CO 80222, (303) 394-0091; FAX: (303) 394-0096.



SLR Systems Upgrades OPTLINK

SLR Systems Inc. has upgraded OPTLINK for Windows. OPTLINK v5.0 for Windows lets developers generate compressed executables (.EXE) and Dynamic Link Libraries (.DLL). According to the company, OPTLINK v5.0 for Windows can reduce file size by 50 percent. Compressed EXEs and DLLs generated by OPTLINK become self-loading and decompress automatically as segments are demanded.

OPTLINK v5.0 supports Borland, Microsoft, and Symantec C++ compilers. Debugging support is included for Borland's Turbo Debugger and Microsoft's CodeView. As a DOS-hosted linker, OPTLINK is distributed in two forms, real mode and protected mode (DPMI).

OPTLINK v5.0 for Windows is \$350. Registered owners of v4.0 will receive the v5.0 update free, while other previous owners can purchase the update for \$165. For more information contact *SLR Systems, Inc.*, 1622 N. Main St., Butler, PA 16001, (412) 282-0864; FAX: (412) 282-7965; BBS: (412) 282-2799.



Segue Releases QA Partner for NT and OS/2

Segue Software has released its QA Partner cross-platform GUI test tool for IBM's OS/2 and Microsoft NT. Using QA Partner, developers can test applications on either of these platforms, and the test scripts will be portable across the other platforms QA Partner supports, including Windows, Macintosh, VMS, and Motif.

QA Partner for both IBM OS/2 and Microsoft NT is \$1,495. For more information contact *Segue Software, Inc.* 1320 Centre St., Newton Centre, MA 02159, (617) 969-3771; FAX: (617) 969-4326.



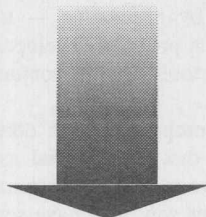
Intel Ships Plug and Play Kits

Intel Corporation has begun shipping three Plug and Play Development kits. The kits are designed to help developers in making PCs easier to configure. The Plug and Play Kits contain the software to perform the automatic configuration of new Plug and Play cards and Plug and Play-ready systems.

The first kit, the "Plug and Play Kit for MS-DOS and Windows," includes a DOS driver, interface libraries, a configuration utility, a VHDL description for an ASIC implementation, and reference manuals. The second kit, the "Plug and Play BIOS Enhancements Kit," contains BIOS software that detects and configures PCI and Plug and Play ISA add-in cards. Software is available in source code form. The third kit, "The Plug and Play ISA Hardware Demo Kit," contains a functional audio Plug and Play ISA demo card and Windows v3.1 Virtual Device Drivers. The kit also includes diagnostics, board schematic files, and speakers. Bundled with this kit is the "Plug and Play Kit for MS DOS and Windows."

The Plug and Play ISA Hardware Demo Kit is \$895. For more information contact *Intel Literature Packet #F8P01*, P.O. Box 7641, Mt. Prospect, IL 60056, (800) 548-4725.

FREE INFORMATION



To receive free product information from the companies listed on this page, simply follow these steps:

- 1) Call 1-800-234-0141 from any phone.
- 2) Follow the voice prompts; be sure to have the product document number and your FAX number handy.
- 3) That's it! The information you've requested is FAXed to you immediately.

It's simple, easy and FREE.

Don't miss the additional information available about *The C Users Journal*. You can receive information about books, back issues, code availability, and much more.

Thank you for using
The C Users Journal
InstantInfo Response Service!

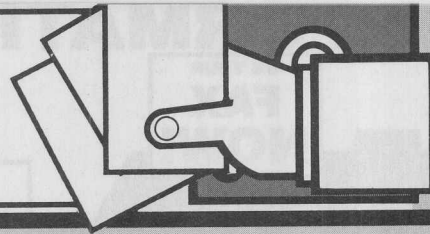
Advertiser	Page	Document
Burton Systems Software.....	31	1186
C Associates.....	140	1156
Cobalt Blue, Inc.	37	1157
Courseware Applications.....	143	1187
DBx, Inc.	54	1195
DC Micro Development.....	138	1151
DMARZ Diversified Sciences Co.	139	1194
EliaShim Microcomputers, Inc.	53	1159
Embedded System Products, Inc.	127	1152
Emerging Technology.....	113	1198
Greenleaf Software, Inc.	CII	1153
Ted Gruber Software.....	18	1163
Intelligent Solutions, Inc.	141	1189
Intertech, Inc.	129	1192
InTrepid Technology.....	123	1154
Just Logic Technologies.....	99	1185
KADAK Products, Ltd.	43	1188
Micro-Digital, Inc.	114	1160
Network Dynamics.....	139	1196
Paladin Software.....	100	1197
Spider Software.....	143	1193
Software Blacksmiths.....	103	1162
Voice Information Systems, Inc.	141	1165
Windbase Software.....	89	1191

Magazine Services	Document
CUJ Subscription Information.....	1167
Code Disk Subscription.....	1168
Code Availability.....	1169
Back Issues.....	1170
CUJ Author Guidelines.....	1171
On-Line Index to <i>CUJ</i>	1172
C Users Group Software Library.....	1173
Windows/DOS Developer's Journal Subscription Information.....	1174
Mailing List Information.....	1175

Book Information	Document
C++ Programming Guidelines (Plum & Saks).....	1176
C Programming Guidelines, 2nd Ed. (Plum).....	1177
Illustrated C (Zolman).....	1178
MS-DOS System Programming, 2nd Ed. (Ward).....	1179
mC/OS The Real Time Kernel (Labrosse).....	1180
Windows Custom Controls (Smith & Ward).....	1181
Object-Oriented Software Engineering (Halladay/Wiebel).....	1182
The Standard C Library (Plauger).....	1183
The C Users Journal Book Catalog.....	1184

Free Product Information — 24 hours a day!
1-800-234-0141

Call for Papers



The C Users Journal is seeking articles on the topics below. If you have an idea for a related story, or experience that would qualify you especially to write on one of these topics, please write:

P.J. Plauger, Senior Editor
The C Users Journal
1601 W. 23rd St., Suite 200
Lawrence, KS 66046-2700
(913) 841-1631

The C Users Journal is written by practicing programmers for practicing programmers. We insist on practical treatments of real programming problems, or on tutorials that make theory or complex issues more accessible to practicing programmers. In place of product reviews, we publish user reports that

show — with insight backed by experience — how to use commercial products that aid the program developer. Our emphasis is on C and C++, or on tools used in conjunction with these languages.

We pay at rates that are competitive with other national technical journals. We believe that our editorial assistance is better than most. You don't have to be a professional writer to meet our acceptance criteria, but you must have something to say. Ask for a copy of our Author Guidelines.

A proposal consists of a brief abstract and outline, preferably totalling no more than one page, and a brief resume of your qualifications. You *must* submit your manuscript on an MS-DOS diskette, or via email (marc@rdpub.com). Files must be in raw ASCII format, flush left, with one empty line between paragraphs.

Themes

Software Tools

Proposals Due 02/07/94

Drafts Due 03/21/94

Suggested topics: portable "shells" and toolkits; small programs that serve as useful building blocks for applications; small programs that aid program development or analysis; reusable tools in a windowing environment; C/C++ libraries that make tools more uniform.

Overworked topics (require fresh perspective): heap trace analyzers, "make" utilities.

Graphics

Proposals Due 03/07/94

Drafts Due 04/18/94

Suggested topics: writing C or C++ code that ports easily among the major GUI environments; useful graphics classes and libraries; algorithms for drawing, shading, windowing, scaling, etc. that are efficient and/or elegant; techniques for visualizing data; manipulating gray-scale and color images.

Overworked topics (require fresh perspective): graphics object classes, graphic file reformatters.

Windows Programming

Proposals due 04/08/94

Drafts due 05/20/94

Suggested topics: C++ class libraries and other scaffolding that simplifies programming under Windows; debugging under Windows; writing C/C++ code that is portable across Windows/Mac/X; useful small Windows utilities in C/C++; using C/C++ to coordinate (script) running other programs; simplifying input under Windows.

Overworked topics (need fresh perspective): simulating stdout or stderr in windows.

Also

Features

If you have an idea for an article that you think might help other C or C++ programmers, you don't have to wait for a more-or-less appropriate theme to roll around. Send it in anytime. Uses for C and C++ are growing too rapidly to count, particularly in a mere twelve categories. So use your imagination, and we may use your words.

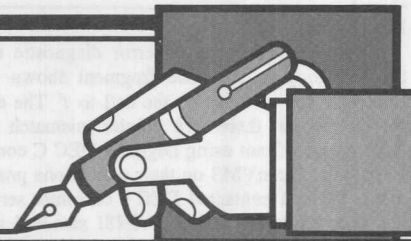
User Reports

If you are willing to share your experiences with a certain product, send us a user report. A user report is not a product review. It is a more anecdotal description written by someone who uses products to get useful work done. Instead of letting vendors trumpet their own products, we want you, the reader, to do it for them — or say why the trumpets should be muted.

Book Reports

If you are willing to share your opinion about a book you have read to improve your understanding of software development, send us a book report. Give your fellow programmers a brief synopsis: an overall evaluation of the book, listing the strong and weak points. Include suggestions for improvement. For more information, ask for the Author Guidelines.

We Have Mail



Letters to the editor may be sent via email to cujed@rdpub.com, or via the postal service to Letters to the Editor, The C Users Journal, 1601 W. 23rd St., Ste. 200, Lawrence, KS 66046-2700.

Dear Mr. Plauger:

There are a number of aspects of the current state of the art of software engineering that I find less than satisfactory. They are

- 1) Bug-infested tools
- 2) The dearth of productive engineering tools
- 3) The programming languages themselves.

First of all, I have used Lattice C, Turbo C, and Microsoft and Borland C/C++ compilers. Our group chose the latter two for their third-party and Windows support. First I used Lattice C 3.1. It didn't do *floats* but *doubles* seemed to work OK. Next I used Microsoft v5.0. It optimized code out of existence. Because of that we switched to Turbo C/C++. It refused to do assignments of floating point (*float* or *double*) literals in the range between 0 and 1!

This problem only occurred when linking together 20-odd modules. In a three-module test program all worked fine. Due to project deadline pressure I was never able to whack away at the 20-odd modules to the point where the problem went away in order to determine its cause. I worked around it. This problem was also present in Borland C/C++ 3.0, but does not occur in version 3.1 with identical source code. I have seen two instances where Borland C/C++ 3.1 handled the following snippet where the function returns the value 1 by not taking the *if*:

```
if ( func() ) // returns 1
    do_something(); // never executed
```

Changing this to the following worked, however:

```
status = func();
if ( status )
    do_something()
```

My first gripe about the state of the art is that compiler vendors seem to concentrate on features like IDEs, 32-bit versions (64-bit next, no doubt), Windows support, etc., while basic functionality is not solid. Floating point seems especially difficult for these people. (You'd think that after ten years of writing C compilers...) I really don't know if Borland C/C++ v3.1 has floating point problems; I just haven't seen any problems so far. Now if Borland and Microsoft are the least-buggy products on the market, I'd hate to see the others. I would love to be able to trust the compiler, but that seems to be wishful thinking. At least we have debuggers to go roto-rooting in the generated assembly code.

More and more vendors are charging for phone support. Now I get to *pay them* for reporting bugs in their products. Borland actually told me to call their 900 number to discuss a bug! No thanks! Along with the bug situation, I wish compiler error and warning messages were more accurate. In many cases I have to ask myself, "Now what does it *really* mean?" I use PC-LINT religiously. To the compiler vendor's credit, error and warning messages have been getting better in recent years.

Now lets add C++ on top of this whole mess. I enjoyed your article, "Programming Language Guessing Games" in *Dr. Dobbs's* (October 1993). I agree of course with your suggestion to "pick a subset of the [C++] language that minimizes surprises, learn it well, and don't stray from it." The problem is that the compiler vendor chooses to support everything that looks like it will make it through the ANSI committee. If they can't get some of these simple things solid (like the function return value example above) how can I trust them to do some of the complicated things you have described for C++? Even if I *do* restrict myself to a subset of the language, it could be that the other features induce bugs in the parts that I am using. The bottom line is that the majority of our time is spent on maintenance. Compiler bugs cost us. Secondly, the support tools available do not help me with software engineering (as opposed to hacking which I define as, "code it first without thought of design, document it later") as I would like. Diagrams are a neat idea, but then we have to translate the diagram into code by hand and translation is where errors creep in. Management might go for the time spent on diagramming if automatic code generators were available. Code generators for Windows screens and other user interfaces are generally available, but I haven't seen any for other parts of a program. A framework generator would be nice. Hardware engineers have schematic capture and printed-circuit-board layout tools including auto routers, but as software engineers, we do analogous things by hand. Also, I've yet to see a decent cross-reference generator after evaluating several. The ones with usable output are way too slow. I use GNU CTAGS, but wish it did more. I know of no general purpose, highly configurable cross-reference generator. We have written our own programs and editor macros to insert comment blocks for functions and modules and for extracting those into a Word document, but so much more could be done. I am beginning to write add-on tools for Codewright, a very extensible Windows programmer's editor. Thirdly, despite some ugly features of C, I like it a lot (more than Pascal, Modula-3, Eiffel, etc.). For example, C overloads the *static* and *void* keywords, allows functions to return pointers to automatic variables, and various other gotchas mentioned in Andrew Koenig's *C Traps and Pitfalls*. (Being able to execute an array of opcodes is a feature?) But for the sa-

dist, C++ is a real treat. Now we have not two but three meanings of *static* and not two but four meanings of *void*!

C and C++ violate many of the accepted guidelines of language design. Many things can be done several different ways, much of it behind the scenes. I shudder whenever I read one of your articles concerning C++, your latest in *The C User's Journal* on the Standard C++ library included (P.J. Plauger, "Standard C: Developing the Standard C++ Library," *CUJ*, October 1993). I know that I cannot avoid learning C++, but hopefully it will be replaced soon with something simpler. We need new paradigms and metaphors to handle advanced concepts. Languages will *have* to become more complicated, but they should be as simple as possible. I like the approach taken by Meyer in Eiffel of including an extensive set of libraries. The less I have to write myself, the more productive I am. Hopefully, libraries supplied with compilers are well thought-out and bug-free. I wish Borland C/C++ came with the library support that Eiffel or NextStep Objective-C does. A collection of third-party classes or libraries does not have the consistency that a single-sourced library does. I have only touched the surface of some current problems in the software industry. What I would like to know is:

- 1) Which in your experience are the least-buggy C/C++ compilers?
- 2) Do you know of a diagrammer w/code generator? What other engineering tools have you found to be helpful?
- 3) Do you know of any cross-reference generator that outputs a list of identifiers including the module name, line number, and function (if any) they were found in, and that is fast?
- 4) What do you think of Modula-3, Eiffel, and IBM's Object REXX? Thank you in advance for your help. Thanks for your excellent articles. Please keep them coming.

BRUCEDICKEY@VAX.MICRON.COM

Bruce Dickey

Micron Semiconductor, Inc.
2805 E. Columbia Rd., MS 892
Boise, ID 83706

Whew! You raise a whole slew of issues. I agree with you almost across the board, except that I am a bit more sympathetic to the plight of all those compiler vendors out there scrabbling for market share in a turbulent industry. I guess that comes from selling compilers myself for ten years. Growing complexity seems always to offset gains in our ability to manage software development and reduce shipped bugs.

I don't know how to answer any of your questions at all well, but I invite other readers to chip in. My opinion of Modula-3 and Eiffel, for what it's worth, is fairly simple. Both offer interesting combinations of features, but neither has quite pulled off the synergy of C or C++. I don't know enough about REXX to comment. — pjp

Dear Editor,

My compiler issues an error diagnostic on the call to *g* in the code fragment shown in Listing 1, but no error on the call to *f*. The error claims that there is a pointer mismatch in the argument. I am using Digital's DEC C compiler under OpenVMS on their new Alpha processor. When I contacted DEC's customer service, they explained that the ANSI standard allows conversion from a pointer to a qualified type to a pointer to a non-qualified type, but does not allow 'ignoring' the qualifier for a pointer to a pointer to a type. Their argument seemed somewhat niggling to me, so I wanted to appeal to a higher authority. How do you think a compiler should handle this code (shown in Listing 1)? Sincerely,

Dick Hile

T and B Computing, Inc.

24 Frank Lloyd Wright Dr.

P.O. Box 302 - Lobby A

Ann Arbor, Michigan 48106-0302

DEC is correct. The distinction may appear niggling to you, but we meant to do it that way when we wrote the C Standard. — pjp

Dear Mr. Plauger,

I would like to add my two cents' worth to Mr. Mike Musielski's comments (*CUJ*, October 1993) on the nature of C as a language. ("Is C a language?" was his opening question.) The short answer to that question is, no, C is not a language. Neither is any programming language. The programming-as-writing analogy is interesting, but like most analogies it is partly true and partly dangerous.

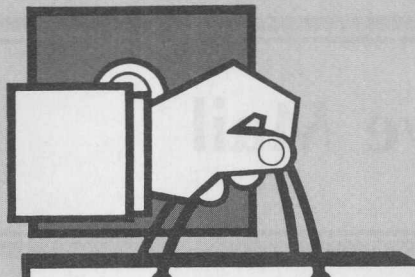
A programming language is really a notation system: defined narrowly, conceived as a formal means of expression, and limited in its range of expressiveness, at least when compared with natural languages. No programming language is used as an end in itself (except of course by participants in Obfuscated C contests), whereas a natural language is used that way all the time by artists whose only goal it is to explore the language's poetic possibilities. This is the main reason why programs, no matter how short, cannot be compared to short stories or other forms of fiction.

Nevertheless, there is a basis for the analogy. Like natural languages, programming languages

Listing 1 Creates "pointer mismatch" error message

```
char x;
char *xp = &x;
char **xpp = &xp;
void f(const char *cp)
{
}
void g(const char **cpp)
{
}
void a(void)
{
}
/* Call to 'g' gets pointer mismatch error */
f(xp);
g(xpp);
}

/* End of File */
```



Programmer's Market

RTCARD™

INTELLIGENT SERIAL COMMUNICATIONS
BY **REALTIME**

Offload Serial Interrupt overhead ... for only \$395

- DUAL PORT
- SYNC / ASYNC
- 8K or 32K RAM
- 18MHz 80C31 μ P
- DMA or I/O
- RS-232/RS-485
- BAUD TO 96K
- 85C30 MPCC

Use our driver software ... or download your own code
Drivers available for:
• DOS • OS/2 • XENIX
• Custom Applications

Ask about our New, Low Cost, Dual Port RS-232 to RS-485 Converter - RTX485

REALTIME CONTROL, INC.
(904) 373-2626
(800) 232-0485

ONLY \$195

◆ Request 191 on Reader Service Card ◆

CRTX EMBEDDED REAL-TIME MICRO KERNEL

- Small Embedded High Performance
- ANSI Standard "C"
- Intertask Communication
- Event Flags (SEMAPHORES)
- Delayed Task Scheduling and Execution
- Timer Services
- Portable
- Compatible With Most "C" Compilers
- Porting and Consulting Services Available

Complete with source for \$99.00

Flanster Software P.O. Box 709, Ramona, CA 92065
(619) 788-5678

◆ Request 258 on Reader Service Card ◆

Toolkit for DOS & Unix

CRUSHER!

DATA & ARCHIVE
MANAGEMENT
COMPRESSION
FOR C, QUICKBASIC & UNIX

LIBRARIES \$189.95 SOURCE \$49.95

Multi-file archives
Subdirectory processing
High compression ratios
Portable archives & source code

Micro Development For free
(606) 268-1559 fax info, call
(606) 266-0726 fax (800) 234-0141
(606) 268-1251 bbs doc #1151

◆ Request 152 on Reader Service Card ◆

New! V3.0 CHIPVIEW-51 The fast path to Turbo Debug 8051 C

- High Level Debugger for all popular 8051 C compilers
- Key Compatible with Borland's popular Turbo Debugger
- Over 14 views including Call Stack and Execution Trace
- **High-Speed Simulator**
Full support for derivatives from Intel, Philips, Siemens, Dallas including all on-chip peripherals
- **Nohau EMUL51 Support**
Both Standard & Advanced Trace, Bankswitch emulator, and Serial box
- **ROM Monitor Version**
Royalty-free source for development as well as in-field debugging

Ask about our Competitive Upgrade Offer
30-day money-back guarantee!
Call for your FREE Working Demo



CHIPTOOLS®
905-274-6244 FAX 905-891-2715

Software Engineers/Programmers

Opportunities exist nationwide for professionals with the following experience:

C, C++, MS/Windows, NT, Unix/Motif, PM, OOP, GUI, Macintosh, SQL, Sybase, Informix, Oracle, or Ingres.

If you have any of the above skills, call or fax us at:



5390 Peachtree Industrial Boulevard • Suite 210-D
Norcross, Georgia 30071 • (404) 368-0711
Fax (404) 368-0713
Fee Paid by Employer

Earn Your High-Tech Degree While Working Full Time

New book *High-Technology Degree Alternatives* shows how to get your college degree without quitting your job, attending night school for years, or breaking your budget. All degree programs are accredited, and many do not require classroom attendance. Use VISA or MC. \$21.95 + \$3.75 s/h.

PPI, Dept. 807
(800) 426-1178

◆ Request 317 on Reader Service Card ◆

IMWHelp



Windows Help Authoring Tool

Professional quality help files
in 1/4 the time!!

- Edit text directly in IMWHelp
- Build hypertext links to: topics, definitions, subjects
- Glossary automatically created
- Bitmaps incorporated
- Desktop publishing features
- Print topics, help file, customized reports
- Spell check, replace verify/all
- Easily reorganize topics, subjects, keywords
- Uses Microsoft Help Compiler

Single User: \$89.95

MC & Visa accepted, Shipping additional
Call: IMCSI (212) 319-1903
425 Madison Ave., New York, NY 10017

◆ Request 170 on Reader Service Card ◆

Low Cost C Source Code to speed up your development

- Decode color TIFF image from HP scanner
- Display as EGA, Gray-Level, & VGA 640x480x256 colors
- Up to 64-level Halftone printing on HP LaserJet & Dot-Matrix
- Production MS C source code for DOS or C/SDK for WINDOWS
- \$59 each, or \$99 for both DOS & WINDOWS
- Custom Programming Available

Order from:

Digital Computing System
17250 West 12 Mile Rd.
Suite 103
Southfield, MI 48076
(810) 557 - 0220
(810) 557 - 0225 (FAX)

◆ Request 122 on Reader Service Card ◆

C++ SOURCE CODE

ImageSoft™

The World's Leading
Publisher of C++
Development Tools
Presents



ISCL™ containing more than 140 volumes of high quality C++ source code (both shareware and freeware) including:

- Neural Nets • Compilers
- Communications • GUIs
- Tutorials • Matrix/Numerical
- Memory Management • Databases/ISAM • Utilities (e.g. Demangler, Beautifier, etc.)
- Graphics • And Much, Much More

All For Only \$99⁹⁹ (on CD Rom)
Also available on 3.5" and 5 1/4"

Contact: ImageSoft Incorporated
2 Haven Avenue • Port Washington, New York 11050
Tel: (516) 767-2233, Fax: (516) 767-9067
EMAIL: medhup@imageliscil

◆ Request 177 on Reader Service Card ◆

TCP/IP

NETWORKING PROTOCOLS

Add them to your
System Designs with:
FUSION Developer's Kit

- FUSION TCP/IP protocol suite
- Flexible architecture - C source code
- Used in hundreds of process control, embedded systems, and end-user designs
- Full support, training, consulting and porting services

(800) 541-9508 (805) 484-2128

Fax (805) 484-3929



Pacific Softworks

◆ Request 494 on Reader Service Card ◆



PROTECT YOUR SOFTWARE TODAY

Since 1986, thousands of companies worldwide have chosen Az-Tech Software as partners in their fight against Software Pirates. Why? Because Az-Tech offers you:

- * DATE & EXECUTION LIMITS* COMPRESSION
- * ENCRYPTION* SERIALIZATION* REMOTE RESET

All of this and now our newest releases in both Hardware and Software based protection:

EVERLOCK 3.0

- * OPTION BOARD SAFE
- * CPU LOCK
- * REMOTE REGISTRATION
- * CD ROM LOCK
- * WINDOWS COMPATIBLE

with over 14 new features you have asked for over the years. If Software based protection is what you need, don't settle for anything less than EVERLOCK 3.0.

These systems are easy to use and most importantly they work! No other product can provide the Compatibility, Flexibility, Ease of Use and Degree of Protection that you receive with Az-Tech products.

EVERKEY LOCKS

We have combined the Versatility of Az-Tech Software and the ONE WIRE Hardware Compatibility of Dallas Semiconductor in this new package.



Call Today For A FREE DEMO
1-800-227-0644



Az-Tech Software, Inc.

201 East Franklin, Suite 11 Richmond, MO 64085
(816) 776-2700 LEADERS IN SOFTWARE SECURITY FAX (816) 776-8398

◆ Request 245 on Reader Service Card ◆

CRYPTO/CRAM™

Data Encryption/Data Compression

Developers Distribution Kit.

Multi-file data encryption with infinite layering!

Build self-Xtracting compressed files.

Distribute self-Xtracting files royalty free!

Compress multiple files easily with a powerful user interface. Utilities included:

CRAM, UNCRAM, Xtract, RUND, RUNK, and DMARZSX. Plus a simple SETUP template.

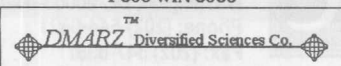
Run executables from CRAMed files.

Create "distribution logs", "security logs", and utilize multiple key access codes.

Entire system is file attribute aware.

CRYPTO/CRAM + distribution kit \$149.00
Shareware version \$30.00

1-800-WIN-8086



◆ Request 310 on Reader Service Card ◆
The C Users Journal — March 1994

DEVELOPERS! ΔΕΓΕΛΟΠΕΡΕ! GOING INTERNATIONAL?

Our Internationalization Toolkit will move
your package to the international market!

- Quickly and easily multi-lingualize your software.
- Add additional user languages with no source code modification.
- Generate Windows string resource files.
- Support double byte character sets, such as Kanji.
- Dynamically switch user languages at runtime.
- Predict and track the growth of strings during their translation.
- Track all information about each string used in your software.
- Insert translated strings back into your source code.
- Maintain your software more easily and reduce your static data space requirements.

NEW! Version 3.0 just released!
More features than ever before!

\$249.95 for C source code and site license. C, C++, BASIC, and Pascal. No royalties. Used world-wide by many Fortune 500 companies on DOS, Windows, Unix, and OS/2 platforms. Complete translation services also available.

Network Dynamics, Inc.

2225 S. Henry Street, Suite L-2
Williamsburg, VA 23185

Phone (804) 220-8771 Fax (804) 220-5741

◆ Request 132 on Reader Service Card ◆

You don't track your bugs?

Save up to 18 fields
per bug, you
choose field
names/values

Automatic
notifications

ad-hoc reports,
graphs, stats

Free demo disk



BugTracker™ \$295

Been There - Done That Software

508-486-9193

FAX: 508-486-8448

◆ Request 315 on Reader Service Card ◆

do evolve and adapt themselves to new situations and conditions. And as they do, new vocabulary creeps in, functions begin to overlap, and things begin to get — well, messy. Even standards cannot clear up all ambiguities: there will always be more than one way to skin a cat. You can say “pass away” or “kick the bucket,” just as you can use *strtol* or *atol*. Both elements in each pair approximate, but do not exactly map, the same idea. How are you to recognize when it is appropriate to use one and not the other? This is what Mr. Musielski wants to know.

There are some useful texts out there. I suggest Mr. Musielski look at (again, if not for the first time) *The C Programming Language*, by Kernighan and Ritchie. Another book I have found useful on occasion is *C Lab Notes*, by Flanders and Holmes. This book has a nice feature in that each chapter has at its head a list of the tasks that are to be explored inside (“forward and reverse scrolling,” “sending a message to another node,” etc.). And, not least, *The C Users Journal* is itself very instructive in matters of language use. None of these texts is consciously “literary,” but all take the implicit stance that a programming language is an idiom to be mastered. Mastering the language (your “material”) is the goal of literary writing as well.

I think what might be very beneficial to Mr. Musielski (and others, myself included) would be a kind of reverse dictionary: a list of common tasks along with the range of functions, procedures, tools, etc. that could be used to fulfill each task's requirement. Such a book could not possibly be exhaustive and the organizational problems would be daunting, but a good text like this might be useful enough to be a starting point. Of course, as writers learn from writing, programmers learn best from programming. There is still no effective substitute for trial and error—many trials, many errors.

Sincerely,

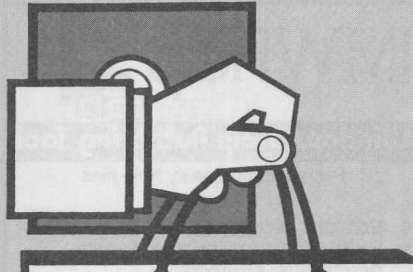
Michael Nichols
1725 York Avenue
New York, New York 10128

Thanks for your insightful comments. — pjp

Editor:

I found Robert Watson's “DMA Controller Programming, in C” (*CUJ*, November 1993) interesting, particularly of course the protected-mode details, but found his conclusion, “the technique is actually very easy to use” laughable, considering what he had just painstakingly documented. “Virtually impossible” is more like it. I agree with his further conclusion: that DMA is increasingly obsolete, since fast CPUs deal with dedicated card memory blocks more expeditiously (and of course the implication that such memory is relatively cheap in modern times — DMA was originally a cost as well as speed feature).

I also think along the way he erroneously minimized the difficulties involved in getting hold of a DMA buffer in real mode (“in real mode, generating a DMA buffer is relatively easy”). I hope I'm missing something, but in my source where I tried to accomplish this there are three or four failed attempts commented-out, and I'm not at all happy with my current effort, shown in Listing 2.




Programmer's Market

International Software Careers

The International Computer Professional Association gives you up-to-the-moment information and world wide contacts you need to find an exciting overseas assignment. Every two weeks you'll receive a detailed listing from international recruiters for software related jobs located throughout the world. Jobs like micro programming/analysis, software engineering, system sales, LAN/WAN support, and others.

As a member of the ICPA you become part of a dynamic international network of software professionals and technical recruiters. People with years of experience in the international software market who will show you how to find an assignment abroad. Call us to receive a membership package for this new global organization of software professionals.

ICPA
350 Townsend Street, Suite 301
San Francisco, CA 94107 USA
24 hour Tel: 1 (415) 695-7618
24 hour Fax: 1 (415) 543-3022



◆ Request 108 on Reader Service Card ◆

SmartHeap

is an ANSI portable malloc and new for DOS, Windows, NT, OS/2, UNIX and other platforms. Proprietary algorithms deliver speed improvements up to 100X versus other commercial mallocs, especially for large heaps in the presence of virtual memory. SmartHeap localizes data structures in their own heaps. Plus it includes numerous fixed-size and handle-based APIs. Debugging facilities isolate leakage, overwrites, double-freeing, wild pointers, and many other errors to responsible file/line/pass count. Exceptionally reliable — used by a “who's who” of commercial software publishers. No royalties. Source available.

CALL FOR FREE WHITE PAPER, BENCHMARKS, AND ERROR REPORTS

MicroQuill
Software Publishing, Inc.
800-441-7822 / 206-525-8218
FAX 206-525-8309
CompuServe: 70751,2443
Internet: devtools@microquill.win.net

ALL mallocs Are NOT Equal

Communications Library

MS DOS and Windows versions.

- Run 4 - 10 ports at once
- 8250 - 16550 UARTs
- RTS - CTS flow control
- DigiBoard PC/4 & PC/8
- Extensive error trapping
- Use any UART address
- Interrupt driven
- COM1 to COM10
- 300 - 115200 baud
- 30+ library functions
- Adjustable queues
- Use IRQ2 to IRQ7

Includes example communications program in C with ASCII, XMODEM, and YMODEM protocols. Get **source** to library, printed User & Reference manuals, quarterly newsletter, one year BBS & phone support. Or download **FREE** demo version from our support BBS (1200 to 9600 baud, 8N1).

Order the **Personal Communications Library** for DOS (PCL4C) \$65, Windows (PCL4W) \$95, or both \$135. Please add \$3 S&H (\$6 overseas). Your satisfaction is guaranteed.

msc
MarshallSoft Computing, Inc.

Post Office Box 4543
Huntsville AL 35815
205 881 4630 Voice / FAX
205 880 9748 BBS
“Reasonably Priced Software Tools”

◆ Request 112 on Reader Service Card ◆

Need to reach 45,000 C/C++ programmers with information about your products?

Advertise in:

The C Users Journal™

Call **913-841-1631** today for information about advertising opportunities in **The C Users Journal**.

Advanced solutions for C/C++ developers.

Brian Osborn - Continental Europe
Ed - East. Christine - Midwest. Edwin - West.


See The Whole Picture.

Let C Associates Expand Your Horizons!.

C Associates will expand your employment horizons as no other UNIX/“C” placement service can. Just call us if you're looking for career advancement in:

- UNIX Systems and Kernel Development • UNIX and C Trainers • Secure UNIX • TCP/IP and X.25 Communications Software • Sybase, Progress, Informix, Ingres, Oracle, and Unify Relational Databases • C & C++ Programmers • System Administrators

Please fax or send resume to:
C Associates
attn: John Capozzi
P.O. Box 15420
Washington, DC 20003-0420
Phone: (202)-544-0821
Fax: (202) 547-8357



◆ Request 137 on Reader Service Card ◆
The C Users Journal — March 1994

Printer Graphics Libraries

PGL ToolKit is a set of easy to use libraries for generating device independent high resolution printer graphics on most popular printers. Provides 80+ functions to create vector drawings and/or print bitmapped images. Provides full control of printing (margins, layout, size, resolution, etc.) and supports parallel and serial port interfaces. Includes full support for C/C++, FORTRAN, Pascal, Basic, Clipper, & Assembly plus 32-bit support for C and FORTRAN.

NO ROYALTIES!

Only \$245 (includes all language support)
Source code: DOS - \$595, UNIX - \$995

AnSoft, Inc.
8254 Stone Trail Court
Laurel, Maryland 20723 USA
Voice/FAX: (301) 470-2335

◆ Request 307 on Reader Service Card ◆

Automated C & C++ Documentation!

Hypertext right from your editor to answer questions like: "What does this function do? Show the code!", "Where is this function?", "What's the name of the class that...?", or "Are there similar functions?".

SourceDoc (formerly PolyDoc) quickly and intelligently scans thru all your source code extracting, creating & categorizing information on functions, classes, definitions, etc. Workgroups have easy access to continuously up-to-date info thru an interactive hypertext view, search, and print engine. SourceDoc also interfaces well with all Version Control Systems including: PVCS, RCS, and TLIB.

Economical Network version available
Intelligent Solutions, Inc.
612/884-0200 FAX: 612/884-5860

◆ Request 184 on Reader Service Card ◆

Is Parsing a BLACK HOLE?

Not when you use *AnaGram*, the new LALR-1 parser generator. *AnaGram* is not just another compiler-compiler — it not only helps you solve your parsing problems, but helps you make them stay solved.

With *AnaGram* you can have recursive parsers, multiple parsers in a single program, multiple instances of a parser, event driven parsers. All with clear, simple interfaces to the rest of your program.

AnaGram's notation simplifies writing and modifying your parsers. *AnaGram* has extensive debugging aids. Even coverage analysis to verify your debugging!

DOS, C/C++. 60-day money back guarantee. Free trial copy available.

800-879-2577 Voice/Fax 508-358-2564
CIS:72603,1763 jholland@world.std.com

AnaGram™ by Parsifal Software
P.O. Box 219, Wayland, MA 01778

◆ Request 300 on Reader Service Card ◆

The C Users Journal — March 1994

Programmer's Toolbox

Dialects: Borland C++; Microsoft C/C++; MPW C & C++; Think C; Turbo C; Zortech C++; Unix C

CLint™ - Syntactically check one or more files & generate ANSI function prototypes; CFlow™ - Determine/graph program organization, function and file interdependencies and runtime library contents; CPrint™ - Reformat/beautify C & C++ source code; CXref™ - Cross reference and check symbol usage and identify unused files; CDecl - Translate and compose C/C++ declaration statements; CHilite™ - Highlight & print C/C++ source files (MPW only); pArc™ - File archiver with great compression (DOS only); pLn™ - UNIX symbolic links in DOS; PMon™ - Program profiler (DOS only); Cpp - flexible ANSI C & C++ preprocessor

30+ productivity improving tools for new code development, software maintenance, documentation, porting, 3rd party integration, ...

Call: (510) 770-0858
MasterCard Visa Accepted
MMC AD Systems
Box 360845
Milpitas, CA 95036

◆ Request 134 on Reader Service Card ◆



Add phone and fax interfaces to your 'C' programs. With the **MULTI-VOICE** and **MULTI-FAX** Toolkits.

- ☎ Multi-Voice for Dialogic, Rhetorex, NewVoice, Bicom, Power Line II, VBX - \$599
- ☎ Multi-Voice for Watson board - \$99
- ☎ Multi-Fax for Intel SatisFAXtion (all models) and other CAS compatible boards - \$199
- ☎ Multi-Tasking Integrated (DESQview version option available - \$99)
- ☎ All Library Source Code Included
- ☎ Many Example Programs



ITI Software

Phone: 514-835-3124

Fax: 514-835-4772

BBS (demos): 514-835-5945

Fax-On-Demand: 514-835-2216

TRANSLATORS

MASM ASM/86

COBOL

PL/I PL/M

to "C"

- Versions for PLM51, 80, 86, 96, 286, 386
- Versions for most Cobol & PL/1 dialects
- Generates "C" source and listing files
- Flags errors; outputs equivalent "C"
- MS-DOS 3.1 +; protected mode option
- Easy to use; includes User's Guide
- From \$475 ea. plus s/h; update service
- Custom translation services available

Micro-Processor Services, Inc.

92 Stone Hurst Lane
Dix Hills, New York 11746 USA
(516) 499 - 4461; Fax: (516) 499 - 4727

◆ Request 162 on Reader Service Card ◆

Phone Sound: Simple!

For Windows/DOS Voice Mail & Fax Developers



Professional Tools for your Voice Mail, Fax & Audiotex Applications

1. Create fantastic prompts and save time with

VFEdit®! Record, crop, cut, copy, paste, mix, fade, echo, volume & more with your Dialogic™ D4x/12x boards.

2. Add Voice Mail power to your MS Windows apps with T1/F DLL™, our Tel I/F Dynamic Link Library.

3. Audio ToolBox™ converts to and from

Multimedia Wave (16, 8 & MS ADPCM), linear 16 & unsigned 8, plus Dialogic 4 & 8 at any sample rate!

4. Scribe Transcription Utility for DOS plays digital audio files in the background without voice mail hardware!

5. Add Text-to-Speech capability to your apps with VaxFonts™, our "software only" text-to-speech library!

Order Now: 1-800-234-VISI

Voice Information Systems: 24 N Merion Ave, Bryn Mawr, Pa 19010
Tel: 215-747-5035/BBS: 215-747-5062/ Fax: 1-800-234-FXIT

◆ Request 238 on Reader Service Card ◆

OPT-TECH SORT/MERGE

New - Version 5

High performance Sort/Merge/Select utility. Run as a stand alone utility or CALL as a subroutine.

Supports most languages and filetypes including Btrieve and dBase. Unlimited filesizes multiple keys and much more.

MS-DOS, Windows \$149
OS/2, UNIX \$249

Call to order or for free info.

Opt-Tech Data Processing

P.O. Box 678
Zephyr Cove, NV 89448
(702) 588-3737

◆ Request 203 on Reader Service Card ◆

Why Can't Documenting Software be as Easy as Writing It!

It Can be by Using

BOCS

The only CASE Tool giving you the freedom and ease you've grown accustomed to when it comes to coding.

☐ Inheritance

☐ Consistency Checking

☐ Automatic Document Generation

☐ "Pretty Printing"

☐ System Level Features

And Lots More

Put an end to documentation drudgery!

Only \$595

Call for a demo copy

Call 301-417-9884 or Fax 301-417-0021.

Berard Software Engineering, Inc.
902 Wind River Ln., Suite 203
Gaithersburg, MD 20878
Email: info@bse.com

◆ Request 216 on Reader Service Card ◆

If anybody knows how to do this rationally, stop me before I code again! The crude strategy here is to keep getting buffers, and if they're no good — won't do DMA because they cross a physical boundary — retain the offending part of the buffer and keep trying; and then when we get a good one, free all the chunks accumulated along the way. This approach relies on totally unreliable assumptions about memory allocation, and really comes down to a "keep trying until you give up" approach.

Operating systems in 80x86 land get to own low memory, where it is relatively easy to set-up a good DMA buffer. Programs, on the other hand, can be loaded at any address, and can't make any such arrangements except, as far as I can figure out, by using awful things like the example.

And incidentally, who owns the VDS (Virtual DMA Services) interface mentioned in the article? I have a vague idea how one goes about getting in touch with DPMI; is VDS one of these things that comes with DOS extenders and/or Windows or what?

j.g. owen
Software engineer
31 Darby Drive
South Huntington, NY 11746
cis 71121,625

Listing 2 Attempts to allocate a real-mode DMA buffer

```
#define BYTE char
typedef unsigned int WORD;
typedef unsigned long BIG;

#define JUNKCHUNK (128)
/* how we will search for physical
boundaries. (Must be >= sizeof(JUNK))
*/

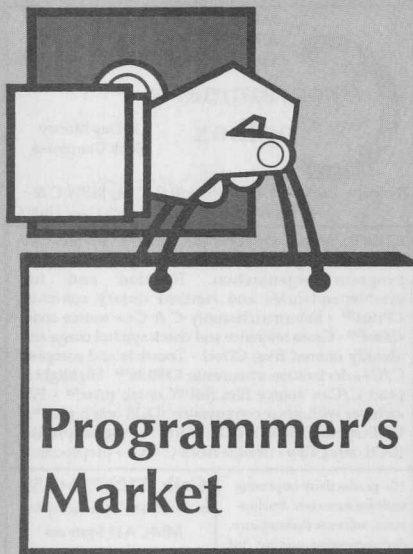
/* produce a big physical address. This
is "large" mode code; " FP_SEG" etc.
are
Turbo C macros which extract the two
parts of an 80x86-style segment-offset
pointer. */

#define PHYSICAL(x) ((BIG)((BIG)
(FP_SEG(x))<<4)+(BIG)FP_OFF(x)))

typedef struct JK
struct JK *next;
} JUNK;

_f void *mustalldma(WORD size) {
BYTE *a;
JUNK root, *next, *p;
BIG leftover;
WORD junksize;
root.next=NULL;
next=&root;

while(1) {
a=mustalloc(size);
/* mustalloc==malloc,
but aborts if failure;
the function exits this
way or via the break below
at success.*/
```



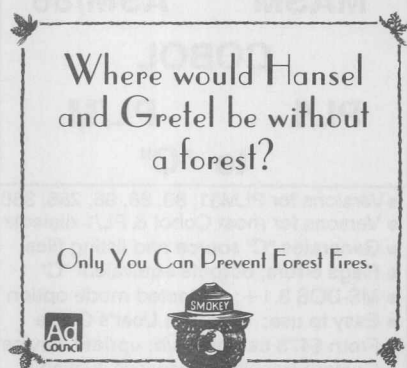
Development Utilities

World's largest/best indexed collections of technical shareware extensively indexed and updated six times a year on CD-ROM or diskettes. 30 day guarantee. Creditcards. Ship/H \$5US, \$20Foreign.

Products	1.44 Disks/Files	Price
Access	12/251	\$59.50
Assembler	12/431	\$59.50
AutoCAD	17/969	\$59.50
C (Turbo & MS)	48/1090	\$149.00
C++ (subset of above)	21/435	\$59.50
dBase & Compilers	65/3307	\$149.00
Turbo Pascal	22/791	\$59.50
Paradox	17/620	\$59.50
Visual BASIC	22/625	\$59.50
ObjectVision	3/100	\$29.50
Netware	72/1176	\$149.00
PC Products Database	137,000 records	\$25.00
TrueType Fonts	15/752	\$59.50
Windows Professional	62/820	\$149.00
Any/All of above on 2 CD-ROMs		\$59.50/\$195

EMS Professional Shareware
4505 Buckhurst Ct.; Olney, MD 20832
(301) 924-3594, Fax: (301) 963-2708

◆ Request 281 on Reader Service Card ◆



USFA Forest Service and your State Forester.

Neural Network Toolkit

Neural network development software for serious applications or for learning:

- ☑ Backpropagation
- ☑ Self-organizing nets
- ☑ Graphical/batch shells
- ☑ C source included
- ☑ Ready to run samples, tools
- ☑ No royalties

Companion \$20 Toolkit \$40
InfiNet \$89



Software Frontiers
Systems, inc.

P.O. Box 8524, Mesa, AZ 85214

1-800-475-9082 Voice/Fax 602-985-8550

◆ Request 282 on Reader Service Card ◆

DIG 3.5 for Windows

Works with Visual C++, BC & NT
NO ROYALTIES

FULL SOURCE

ALL for \$499.00

WAKE UP to the possibilities of advanced scientific graphics in your own software. Save years of programming time and effort.

Get full source code for advanced graphics without paying any royalties. Do not get in the position of negotiating royalties after spending the time to learn and incorporate a graphics library into your program.

Get powerful and robust source code that is easy to understand and modify. Write to any windows device context, including windows, printers, icons, bitmaps and metafiles. Copy metafiles and bitmap graphics to the windows clipboard for export to other applications.

Get all the advanced graphics that you want: xy, error, Smith, bar and pie charts. Get solid or wireframe surfaces with color coded height and solid or labeled line contours, even contours on spheres!

Call (619) 632-9510 & Order It.

CHIRP Technical
Services

P.O. Box 551, Del Mar, CA 92014

The Only Optical Form Recognition

Form Scan

Recognizing:

Text: Position, Point size and Attribute
Line, Box: Position, Length and Weight
Field: Position, size, Left or Right
alignment for filled-in data


Output: All out application to display
and fill (with source) and formats in:

- C: Make file, source file
definition file, header file
and resource file
- Visual Basic: Form description
file and project file
- Jetform and others

E-Data Link: (713) 690-5710

◆ Request 181 on Reader Service Card ◆

The C Users Journal — March 1994



Drawbridge

Source Code Generating Graphics Editor

Spare yourself the tedium of programming graphics. Use Drawbridge to create:

- logos,
- screen prototypes,
- graphics,
- animations,
- demonstrations, and
- instructional applications.

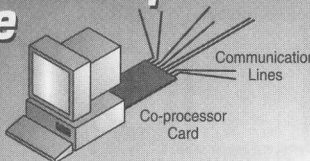
Includes support for Borland BGI, Genus GX, MetaWindow, and Microsoft graphics libraries. 30-day money back guarantee. Demonstration disk available.

Special! \$12900

Courseware Applications
One Appletree Square, Suite 989 • Bloomington, MN 55425
Tel: (612) 854-8909 • Fax: (612) 854-1868

◆ Request 130 on Reader Service Card ◆

Cut development time



X.25 LAP-B HDLC SDLC Bisync & Custom Protocols

Team up with Quadron's C-language development environment to offload communications to IBM's ARTIC realtime co-processor cards. We'll supply OS/2 or DOS services & tools to write multi-tasking applications for your system unit and cards. Development time is short because we handle low-level ARTIC code for you. You'll use standard compilers and C library functions, and debug quickly with our realtime symbolic debugger & statistics monitor. Call now.

Quadron 805-966-6424
209 East Victoria Street, Santa Barbara, CA 93101

◆ Request 291 on Reader Service Card ◆

DICE IS LOOKING FOR...

...data processing, engineering and technical writing professionals to fill open positions nationwide. DICE is a FREE online job search service providing detailed information about current contract and full-time positions across the USA. It's a confidential, easy to use, no cost way to search for a new job.



**DATA PROCESSING
I NDEPENDENT
CONSULTANT'S
E XCHANGE**

ONLINE Number 515-280-3423

Contact DICE via 1200/14400 baud Modem, 8-N-1.
A service of D&L Online, Inc.
515-280-1144

◆ Request 301 on Reader Service Card ◆

NETWORKING PROTOCOLS

- ◆ TCP ◆ ISDN
- ◆ X.25 ◆ Frame Relay*
- ◆ OSI ◆ ATM*

- ◆ Portable STREAMS-based protocol stacks
- ◆ All software written in C
- ◆ Full support, training, porting and consultancy services
- ◆ Source code licensed to more than 80 OEMs and Systems Integrators

Spider Software

For Further Information call us now on:
Tel: 415-546-0606 or FAX: 415-543-5611
Tel: 508-460-0049 or FAX: 508-460-0107
(*Available during 1994)

◆ Request 323 on Reader Service Card ◆

C PROGRAMMERS: BRIGHTEN YOUR FUTURE!

Positions available in all phases of product development working for a leader in the LAN Enhancement Software Industry. C/C++ programmers needed to develop NetWare NLMs and Protocols. We offer:

- Excellent Starting Salary
- Full Benefit Package w/ 401K
- Flexible Hours
- Central New Jersey Shore Location

Please call or send resume today to
Michael Randazza
Brightwork Development, Inc.
Jerral Center West
766 Shrewsbury Ave.
Tinton Falls, NJ 07724
908-530-0440 x4523
908-530-0622 fax



◆ Request 269 on Reader Service Card ◆

BAS_C

BASIC to C translator

- inputs **BASICA**, **Quick BASIC**
- outputs **MS/Quick/Turbo C** [++]
- **CUSTOMIZER™** for other BASIC
- **restructures any spaghetti code**
- **indented, scoped, modularized C**
- **syntax-error free, portable C**
- **runtime library written in C**
- **run on MSDOS, XENIX, UNIX**
- **1,000 copies sold since 1986**
- **FREE DEMO DISK, INFO**
- **Call for non MSBasic**

Starting from \$599
Gotoless Conversion
7105 Dee Cole Drive, The Colony, TX 75056
Phone (214) 625-2323
Fax: 214-370-2612 BBS: 214-625-6905



THE TREASURE HUNTERS

In the world of software, it is all too easy to lose a valuable treasure: by editing and saving a program, overwriting a vital previous version.

To prevent you from having to search in vain for buried treasures, **ARIS VERSION TRACKING SYSTEM** for Windows enables you to access and use previous program versions.

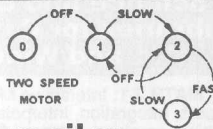
AVTS. A gem of a program at a diamond price.

ARIS Information-Systems
Parkstr.2, 85646 Anzing, Germany
Phone +49-8121-45624, Fax +49-8121-45625
From the USA, call
Phone 01149-8121-45624
Fax 01149-8121-45625
Email: CompuServe 100042,1707



◆ Request 135 on Reader Service Card ◆

FINITE STATE CASE Compiler



The **COMPEDITOR**, a CASE software development tool, quickly forms and documents event driven finite state, automations, and decision table programs. Forms state tables (to 11K cells) and the program's 'C' source code.

Determine what the program should do when the conditions represented by a state table cell occur. Enter in the cell the code or function(s) to call and the program's next state. Price \$300

Send for our free Tutorial

AYECO, Inc. (407) 295-0930
5025 Nassau Circ, Orlando, FL 32808

◆ Request 399 on Reader Service Card ◆

C and C++ DOCUMENTATION

! AUTOMATED DOCUMENTATION !

- **C-CALL (\$69)** Graphic-tree of caller/called functions, cross-ref, file/function index.
- **C-CMT (\$69)** Creates/inserts/updates comment-blocks for each function, listing the functions and identifiers used by it.
- **C-METRIC (\$59)** Counts path complexity, counts comments, code, 'C' statements.
- **C-LIST (\$69)** Lists and action-diagrams, or reformats into standard formats.
- **C-REF (\$59)** Creates cross-reference of local/global/define/parameter identifiers.
- **SPECIAL: C-DOC (\$199)** All 5 programs integrated as DOS program (<15,000 lines)
- **NEW! C-DOC Professional (\$299)** DOS, OS/2, Windows. 3-ring binder/case. Processes 150,000 lines, deferred reports.
- **30-DAY Money-back guarantee CALL NOW**

SOFTWARE BLACKSMITHS INC.
6064 St Ives Way, Mississauga
ONT, Canada Voice/Fax (416) 858-4455
LSN-4M1 Demos/BBS (416) 858-1916
see AD INDEX for our larger ad

◆ Request 142 on Reader Service Card ◆

To the Editor:

Concerning the letter from Lawrence H. Hardy in the 11.12 issue of *CUJ*, page 136, asking for information about PCX graphics:

And I thought my memory was going... Actually, this might instead be seen as an illustration of the need for a *CUJ* index — depending on one's perspective of course. At any rate, whilst fumbling through a stack of old *Journals* seeking edification on a particular coding problem, I came across Vol. 9, No. 8 (August, 1991). Emblazoned across the cover was "PCX GRAPHICS DOCUMENTED!" And sure enough, there on page 89 was a decent discussion of the topic by one Ian Ashdown, complete with bibliography.

I would also like to join Ian Somerton (p. 127) in expressing concern, if not alarm, at the exponential increase in C++ coverage of late. Perhaps someone on your staff misread it as C**?

Sincerely,

Scott Swanson
Box 75
Pendroy MT 59467

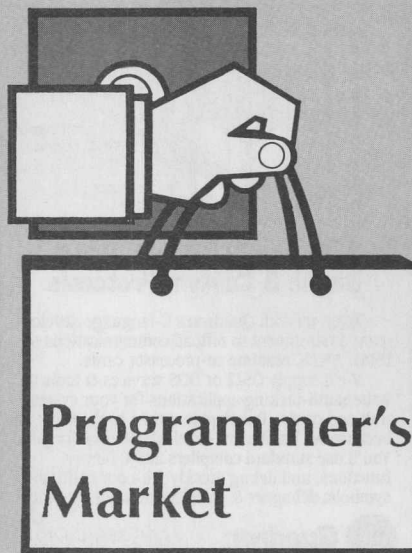
I am notorious within R&D for my instant amnesia, once an issue goes out the door. You're right that an index to CUJ is indispensable, and the folks back in Kansas have anticipated your wish. As for C++ coverage, that happens to be an area of intense interest in our community at the moment. We haven't forgotten about C by any means, but the mix what we publish is strongly influenced by the mix of what's proposed to us in the way of articles. — pjp

Listing 2 continued

```
if (
(leftover=(PHYSICAL(a) & 0xffff)) +
((BIG) size)
>= 0x10000) {
/* it's no good. */
if (
(junksize = 0x10000-leftover)
< JUNKCHUNK)
junksize=JUNKCHUNK;
/* avoid endless
teeny-tiny manipulations. */

free(a);
next->next=malloc(junksize);
next=next->next;
}
else
break; /* success! */
}
/* free debris. */
next=root->next;
while(next)
p=next;
next=next->next;
free(p);
}
return a;
}

/* End of File */
```



REAL TIME Multi-Tasking Operating System

- Task Manager
- Queue Manager
- UART Manager
- Resource & Semaphore Manager
- ROMable
- User Configure-able
- All CMX Functions Contained in a Library
- Preemptive Scheduling
- Event Manager
- Cyclic Timers Manager
- Message Manager
- Memory Manager
- Supports Nested Interrupts
- All Functions Written in "C" for Portability
- Task Scheduler & Interrupt Handler in Assembly for Speed and Optimization

The NEW CMX-TINY is also Available

ST9 Family	7700
80C166/167	78K2/K3
8051 and Derivatives	8096/196
H8/3XX, H8/5XX	68HC11/16
Other Microprocessors	Z80/64180

Source Code No Royalties

CMX 508-872-7675
Company FAX: 508-620-6828

◆ Request 325 on Reader Service Card ◆

Science and Engineering Libraries for C, C++, Pascal, and Windows

G_MATH 3.1: Interactive Math, Real & Complex Roots, Integration, Interpolation, Matrix, Curve Fitting, graphics, & more. **\$149.**

G_FFT 3.1: Interactive Signal Processing, Real, complex FFT, Correlation, Convolution, Power Spectrum, Filter Design, Graphics, & more. **\$149.**

G_FFT 3.1: Class Library. **\$149**

OOPlot: Interactive Scientific plotting, Linear, Log, LogLog, Scatter, Bar, Pie graph, grids, Origin setting, overlapping windows, & more. **\$149.** Class Library: **\$149.**

OOParser Class Library, Expression and function evaluator with 1, 2, & 3 variables and unlimited parameters. **\$79**

30-day money back. No Royalties. Free Technical Support.

Sigma Software

15779 Columbia Pike, suite 360
Burtonsville, MD 20866

**Call or Fax your order to
301-317-6207**

◆ Request 114 on Reader Service Card ◆

Developers:

For the next couple of months we will be telling you about a new set of shareware utilities called:

VSST: Valois Software System Toolkit

This month we are featuring **VHDISK:**

- >> Identifies IDE drive manufacturer & model
- >> Lists bad sectors not listed in DOS FAT, and how many spare sectors left before data loss
- >> Modify cache/error configuration settings
- >> Performs many seek and data transfer tests
- >> Allows power control over spindle motor

Unlike other utility packages, **VSST** can be run in *batch mode*, not only displays but *saves* the information to disk so that it can be used later for *reporting/comparison* with other systems, and comes with .LIB files: *link with your code!*

Call Toll-Free: 1-800-884-SOFT

Valois Software Inc.

2237 Cecilia Ave. San Francisco, CA 94116
Fax: (415) 759-0948 CIS: 71117,1762

◆ Request 287 on Reader Service Card ◆

Only
\$395

Fuzzy Logic Designer™

The tool to use for applied fuzzy logic,
not hazy theories

Now with neural rule weights

- Generate portable C source, unique to your design
- Includes detailed documentation and tutorial explaining fuzzy logic theory
- Use any debugger, emulator with your application or embedded system
- Provides graphical simulation capability and GUI using Microsoft Windows(tm)
- No royalty or license fees for generated output
- Only \$395.00. Call for information today.
- Additional OEM and DLL capability available



Byte Dynamics, Inc.

14608 E. Olympic Ave.
Spokane, Wa. 99216
Call: (800) 233-2983
Fax: (509) 926-6130

* Custom software development available *

◆ Request 125 on Reader Service Card ◆

PostScript From C!

c_pslib

- ✓ Complete Set of Graphics Functions
- ✓ Text Functions for fonts, lines, paragraphs
- ✓ Downloadable Font support
- ✓ Encapsulated PostScript File support

c_psFontInfo

- ✓ Stringwidth Calculations
- ✓ Font Reencoding
- ✓ Kerning, Ligatures, & More!

Site License \$595 each

Single User License \$195 each

ANSI and K&R Source Code Included

C_Graph Software Inc.

P.O. Box 5641 Austin, TX 78763

512-444-3336

◆ Request 261 on Reader Service Card ◆

The C Users Journal — March 1994

C, C++ and BASIC programmers, now you get much more than xBase compatible DBMS power.

Thousands of programmers have already discovered how to get dBASE, FoxPro and Clipper compatibility from their favorite language and hardware platform. For example, one customer has C programs running on PC and Sun workstations sharing data with concurrently running FoxPro for Windows applications. You see, CodeBase technology is simply the best way to add multi-user xBase compatible DBMS power to C, C++, or BASIC.

You still get high speed & small size CodeBase users really appreciate our small executable size. Unlike SQL engines which are a Meg or so in size, CodeBase 5.1 EXE's can be as small as 45K! You'll also like the speed—with our Intelligent Queries you get the execution speed of C plus stunning query performance from our smart use of available index information.

Introducing CodeControls

Now formatted data entry in Windows is as easy as point & click!

Introducing the new CodeControls, a unique set of data-aware custom controls. Now simply drop them into your Windows applications via your favorite visual interface builder.

NEW—You get formatted data entry
Experienced Windows programmers know **formatted** data entry is difficult

to program under Windows. But with our new **CodeControls**, you can simply 'Point & Click' to design data entry windows for date, numeric, and character information—formatted just the way you want it.

NEW—Data-aware controls

Our custom controls are **data-aware**, so now you can easily build a scrolling list box that's tied to a data file, or look up matching combo box entries—even as the user types.

Introducing CodeReporter 2.0

Now use the new Instant Report Wizard to create a variety of reports—instantly.

Introducing the new CodeReporter 2.0, our interactive xBase report writer. We designed it with developers in mind, but end-users will love it.

NEW—You get instant reports

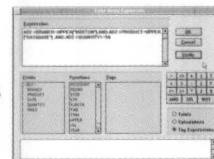
Use our new Instant Report Wizard to quickly create a wide variety of reports—in an instant, then easily refine them as necessary, or let your end users build their own special reports!

NEW—You get drag & drop

We've added drag & drop capabilities to **CodeReporter**, so regardless of whether you want data, totals, or text, simply drop a report object onto your layout screen.

NEW—You get interactive queries, sorts, & relations

Use CodeReporter to visually design reports easier than ever. For example, you can quickly build query and sort expressions using our calculator-style expression builder. In addition, you can easily relate data files together using our graphical relation builder.



"There's never been a better time to buy into CodeBase technology. You get complete xBase compatibility, full DBMS power, plus an interactive report writer and a set of unique data-aware custom controls for Windows."

—Ken Sawyer, President, Sequiter Software

Buy One, Get Two FREE.

Now when you buy any one of our xBase library products: **CodeBasic**, **CodeBase++**, or **CodeBase** (for the language of your choice), you'll get both the new **CodeReporter 2.0** AND the new **CodeControls** absolutely **FREE**—for a limited time only.

**To Order Now Call
403-437-2410**

Unconditional 90-Day Money-Back Guarantee

SEQUITER SOFTWARE INC. FAX 403-436-2999
UK Tel. +44-81-317-4321
France +33.20.24.20.14

P.O. Box 575 Newmarket NH 03857-0575

©1993 Sequiter Software, Inc. All rights reserved. CodeBase, CodeBase++, CodeBasic, CodeReporter, and CodeControls are trademarks of Sequiter Software Inc. All other product names mentioned herein are trademarks of their respective companies.

FAST, PORTABLE C DEVELOPMENT TOOLS

✓ Client/Server

✓ Proven Performers

✓ Develop Royalty-free
with c-tree

✓ Unsurpassed Portability

✓ Source Code

✓ Customizable

✓ Powerful Report
GenerationA base
B base
C base
...
X baseSD '94
Stop by our
booth #1744More
CapableMore
AffordableExpensive
DBMS Servers\$ FAT MAINTENANCE \$
Heavyweight AirwaysLightweight Solutions lack features
and flexibility you may need
to accomplish your mission.Heavyweight Solutions will cost
a small fortune before
you even leave the ground.FairCom® Offers A Full Line Of High-Performance Servers
& Development Tools At Reasonable Prices.FairCom Servers®
SQL & non-SQL

These multi-threaded database servers offer a seldom-found solution for developers who demand control. While one may mandate SQL access, another's real-time demands may not tolerate the overhead associated with SQL. Our unique servers offer the full range of data accessibility: low-level speed; convenient ISAM-level; compatible SQL-level. Complete **data integrity** is achieved with multi-user transaction processing. Recovery of all committed transactions after a failure is fully automatic. The C developers 'CLIENT/SERVER' solution: **DOS nodes to UNIX servers**; full commit and rollback; roll forward; precise control over your data and/or data base model; large files (4GB); **complete client source code**. On-line data backups with maximum data availability and protection. These servers are designed to achieve maximum power and flexibility without sacrificing control.

r-tree®
Report Generator

r-tree report generator provides complex, multi-line reports by virtually handling every aspect of report generation. Your only programming requirement is to call the r-tree report function, which reads c-tree data files, performs calculations, monitors control breaks and accumulators and produces a formatted report.

c-tree Plus®
High Performance Data Management

Based on the most advanced B+ Tree routines available today, c-tree Plus gives you unprecedented control over your file management needs. With unparalleled sophistication, c-tree Plus has established itself as the premier choice for commercial development. Use the low-level routines or take advantage of the high-level ISAM routines for high speed random or sequential access. c-tree Plus is distributed in **complete C source code** and is known for its portability and **royalty-free** licensing policy. Whether for single-user, multi-user or client-side application development, c-tree Plus delivers. Transaction processing is included in c-tree Plus as well as features like: fixed/variable length records and keys; dynamic space reclamation; high speed hashed data and index caching (and savepoints, abort, and rollback); full ISAM functionality; and batch operations. All functions are fully ANSI prototyped. Call for a complete list of features.

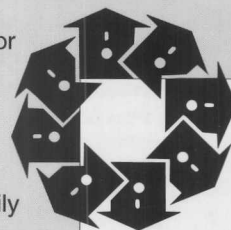
Natural Query™
Natural Language Tool

Produces ad hoc reports quickly and easily from English sentences. It is the first **low-cost**, natural language tool designed for developers and a broad range of users. The QBE option has **easy-to-use**, pick and choose menus, making report specifications a snap. Available in VAR or end-user versions.

✦ Request 128 on Reader Service Card ✦

d-tree Toolbox®
Application Development

Productivity tools with: a complete **portable screen handler**; data dictionary; code generation; easy to use data base interface; menus; help text; data validation. d-tree's dynamics allow **runtime control** of program resources (screens, files, edits, etc.) not found in any other development package. Resources can be changed in memory, and/or swapped on/off of disk at runtime. Additional productive features include: flexible data windows; field masks; user-defined and field-level edits and **data file reformatting**. If you do application development on multiple platforms (DOS/UNIX), or you're debating 4GL versus C development, d-tree is for you.

Call today for more information:
(800) 234-8180FAIRCOM®
corporation4006 W. Broadway · Columbia, MO 65203
(314) 445-6833 Fax (314) 445-9698FAIRCOM EUROPE
Via Sottocorna 15/17 · ALBINO (BG) ITALY
tel. 035/773464 · FAX 035/773806